MATHIEU NEBRA

RÉALISEZ VOTRE SITE WEB AVEC HTML5 ET CSS3

CRÉER SON SITE WEB N'A JAMAIS ÉTÉ AUSSI FACILE !



Issu du célèbre
Site du Zéro
www.siteduzero.com



www.siteduzero.com

MATHIEU NEBRA

RÉALISEZ VOTRE SITE WEB AVEC HTML5 ET CSS3

CRÉER SON SITE WEB N'A JAMAIS ÉTÉ AUSSI FACILE !



www.siteduzero.com

DANS LA MÊME COLLECTION



APPRENEZ À PROGRAMMER EN C

MATHIEU NEBRA
ISBN : 978-2-9535278-0-3



CONCEVEZ VOTRE SITE WEB AVEC PHP ET MYSQL

MATHIEU NEBRA
ISBN : 978-2-9535278-1-0



RÉDIGEZ DES DOCUMENTS DE QUALITÉ AVEC LATEX

NOËL-ARNAUD MAGUIS
ISBN : 978-2-9535278-4-1



REPRENEZ LE CONTRÔLE À L'AIDE DE LINUX

MATHIEU NEBRA
ISBN : 978-2-9535278-2-7



APPRENEZ À PROGRAMMER EN JAVA

CYRILLE HERBY
ISBN : 978-2-9535278-3-4



PROGRAMMEZ AVEC LE LANGAGE C++

M.NEBRA ET M.SCHALLER
ISBN : 978-2-9535278-5-8



RÉDIGEZ FACILEMENT DES DOCUMENTS AVEC WORD

MICHEL MARTIN
ISBN : 978-2-9535278-7-2



APPRENEZ À PROGRAMMER EN PYTHON

VINCENT LE GOFF
ISBN : 979-10-90085-03-9



DÉBUTEZ DANS LA 3D AVEC BLENDER

ANTOINE VEYRAT
ISBN : 978-2-9535278-9-6

MATHIEU NEBRA

RÉALISEZ VOTRE SITE WEB AVEC HTML5 ET CSS3

CRÉER SON SITE WEB N'A JAMAIS ÉTÉ AUSSI FACILE !



www.siteduzero.com



Simple IT 2011 - ISBN : 978-2-9535278-8-9

Avant-propos

Voilà plus de vingt ans que le Web existe ! Un âge respectable dans l'univers de l'informatique : le début des années 1990 fait presque figure d'Antiquité.

Le Web n'a pourtant jamais été autant utilisé qu'aujourd'hui : nous avons tous pris l'habitude de naviguer sur des sites web, un peu comme si c'était une seconde nature. C'est même devenu notre activité principale sur un ordinateur, à tel point qu'on en vient à oublier qui celui-ci peut aussi servir à autre chose qu'à naviguer sur le Web. Même quand nous ne sommes plus devant un ordinateur, nous avons pris l'habitude de regarder compulsivement notre téléphone portable pour consulter les dernières nouvelles. Bref, le Web nous poursuit partout.

Alors, vous vous dites : « Tiens, si je créais moi aussi mon site web ? Tout le monde le fait, je dois en être capable moi aussi. »... et vous avez parfaitement raison ! Si le Web s'est autant démocratisé, c'est aussi parce qu'il est facile de créer un site web. Mais attention, cela ne s'improvise pas : il faut de la méthode, de l'organisation et quelques connaissances sur les langages du Web. Autant d'éléments que je souhaite vous apporter à travers ce livre !

Vous n'avez jamais entendu parler de HTML ni de CSS ? Pas grave : cet ouvrage est conçu pour les débutants. Tout ce que je vous demande c'est de savoir allumer votre ordinateur (et utiliser une souris et un clavier, tant qu'à faire). Pour le reste, laissez-moi vous montrer tout simplement !

HTML5 et CSS3, ces incontournables langages

Quoi que vous fassiez, vous ne pourrez pas les éviter : pour créer un site web, vous devez apprendre à manier les langages HTML et CSS. Ils sont à la base de tous les sites web, qui sont tous conçus de la même manière avec ces langages.

Les toutes dernières versions de ces langages (HTML5 et CSS3) nous offrent aujourd'hui des possibilités étonnantes. Des effets graphiques autrefois complexes à réaliser se créent désormais via une simple ligne de code. Les langages deviennent tellement puissants que les experts sont déjà d'accord pour dire que demain tout se fera sur le Web. C'est d'ailleurs déjà le cas quand on y pense : courses en ligne, déclaration d'impôts, lecture des journaux...

Vous vous dites peut-être que vous n'êtes pas faits pour apprendre un langage informatique, mais pas de panique : les langages HTML et CSS sont *simples*. Je vois d'ailleurs régulièrement des jeunes les apprendre dès l'âge de 10 ans, comme s'il s'agissait d'une simple formalité. Ce n'est en fait pas plus compliqué que la grammaire et la conjugaison, c'est même parfois plus facile. ;-)

Qu'allez-vous apprendre en lisant ce livre ?

Nous allons découvrir HTML et CSS de façon progressive au cours de cet ouvrage, composé des parties suivantes :

1. **Les bases de HTML5** : nous commencerons par découvrir le langage HTML. Nous partirons vraiment des bases : quels programmes dois-je installer ? Comment tester le résultat de mon site ? Comment afficher du texte, des images, et créer des liens ? Nous verrons tout cela au cours de cette première partie très simple.
2. **Les joies de la mise en forme avec CSS** : nous introduirons ensuite le langage CSS, qui permet de mettre en forme son site web. Nous apprendrons à choisir les couleurs de notre site, à modifier la taille du texte, la police... et même à jouer avec les effets de transparence, les ombres et les arrondis que nous offre aujourd'hui CSS3.
3. **Mise en page du site** : nous passerons aux choses sérieuses en nous intéressant à la mise en page de notre site. Dimensions, marges, positionnement au pixel près : nous découvrirons tout ce qu'il faut savoir. Nous terminerons cette partie en apothéose avec la réalisation de A à Z de notre premier site web complet. Vous allez être bluffés par ce que vous êtes capables de faire !
4. **Fonctionnalités évoluées** : nous irons encore plus loin en découvrant des fonctionnalités plus complexes de HTML et CSS. Nous passerons notamment en revue les tableaux, les formulaires, la gestion des vidéos... Et je vous ferai faire un petit tour d'horizon de tout ce que vous pourrez encore découvrir pour satisfaire votre soif de connaissances !

À la fin de cet ouvrage, plusieurs annexes vous attendent pour compléter ce que vous avez appris : vous verrez notamment comment procéder pour envoyer votre site sur le Web !

Comment lire ce livre ?

Suivez l'ordre des chapitres

Lisez ce livre comme on lit un roman. Il a été conçu pour cela.

Contrairement à beaucoup de livres techniques où il est courant de lire en diagonale et de sauter certains chapitres, il est ici très fortement recommandé de suivre l'ordre du cours, à moins que vous ne soyez déjà un peu expérimentés.

Pratiquez en même temps

Pratiquez régulièrement. N'attendez pas d'avoir fini de lire ce livre pour allumer votre ordinateur et faire vos propres essais.

Utilisez les codes web !

Afin de tirer parti du Site du Zéro dont ce livre est issu, celui-ci vous propose ce qu'on appelle des « codes web ». Ce sont des codes à six chiffres à saisir sur une page du Site du Zéro pour être automatiquement redirigé vers un site web sans avoir à en recopier l'adresse.

Pour utiliser les codes web, rendez-vous sur la page suivante [1] :

`http://www.siteduzero.com/codeweb.html`

Un formulaire vous invite à rentrer votre code web. Faites un premier essai avec le code ci-dessous :

▷ Tester le code web
Code web : 123456

Ces codes web ont deux intérêts :

– ils vous redirigent vers les sites web présentés tout au long du cours, vous permettant ainsi d'obtenir les logiciels dans leur toute dernière version ;
– ils vous permettent de télécharger les codes sources inclus dans ce livre, ce qui vous évitera d'avoir à recopier certains programmes un peu longs.

Ce système de redirection nous permet de tenir à jour le livre que vous avez entre les mains sans que vous ayez besoin d'acheter systématiquement chaque nouvelle édition. Si un site web change d'adresse, nous modifierons la redirection mais le code web à utiliser restera le même. Si un site web disparaît, nous vous redirigerons vers une page du Site du Zéro expliquant ce qui s'est passé et vous proposant une alternative.

En clair, c'est un moyen de nous assurer de la pérennité de cet ouvrage sans que vous ayez à faire quoi que ce soit !

Remerciements

Je souhaite remercier un certain nombre de personnes qui, de près ou de loin, ont contribué à la naissance de cet ouvrage :

– Ma famille, qui continue de me témoigner sa confiance et qui suit toujours avec attention mes projets.
– Élodie, pour son soutien indéfectible. Cet ouvrage t'est dédié en souvenir de cette magnifique journée d'été.
– Pierre Dubuc, qui continue à soulever des montagnes pour faire vivre notre projet commun.

1. Vous pouvez aussi utiliser le formulaire de recherche du Site du Zéro, section « Code web ».

- Notre infographiste, Fan Jiyong, pour la couverture de ce livre, les illustrations des chapitres et le webdesign du TP (qui est plusieurs crans au-dessus du précédent !).
- Notre éditeur, Jonathan Baudoin, pour son implication et son travail efficace lors de la préparation de cet ouvrage.
- L'ensemble de l'équipe de Simple IT qui travaille avec moi sur le Site du Zéro.
- Johann Pardanaud (Nesquik69) et Julien Villetorte (delphiki), pour leur relecture et leurs conseils avisés.
- Vous tous, chers lecteurs, avec qui je prends toujours autant de plaisir à partager mes connaissances. Bonne lecture !

Sommaire

Première partie

Les bases de HTML5

Chapitre 1

Comment fait-on pour créer des sites web ?

Difficulté :

Bonjour et bienvenue à toutes et à tous ! Voici donc le premier chapitre de ce livre pour débutants, qui va vous apprendre à créer votre site web !

Nous allons passer un certain temps ensemble, tout dépendra de la vitesse à laquelle vous apprendrez. Si vous lisez ce livre régulièrement et à une bonne vitesse, vous l'aurez terminé en une à deux semaines. Mais si vous avez besoin d'un peu plus de temps, ne vous inquiétez pas : le principal est que vous y alliez à votre rythme, de préférence en prenant du bon temps.

Je vous propose de commencer par la question la plus simple mais aussi la plus importante : **comment fonctionnent les sites web ?**



Le fonctionnement des sites web



Comment fonctionnent les sites web ?

Non, n'ayez pas peur de poser des questions même si vous pensez qu'elles sont « bêtes ». Il est très important que nous en parlions un peu avant de nous lancer à fond dans la création de sites !

Je suis certain que vous consultez des sites web tous les jours. Pour cela, vous lancez un programme appelé le navigateur web en cliquant sur l'une des icônes représentées à la figure 1.1.



FIGURE 1.1 – Les icônes des navigateurs web les plus répandus

Avec le navigateur, vous pouvez consulter n'importe quel site web. Voici par exemple un navigateur affichant le célèbre site web Wikipédia (figure 1.2).



FIGURE 1.2 – Le site web Wikipédia

Je suis sûr que vous avez l'habitude d'utiliser un navigateur web ! Aujourd'hui, tout le monde sait aller sur le Web. . . mais qui sait vraiment comment le Web fonctionne ? Comment créer des sites web comme celui-ci ?



J'ai entendu parler de HTML, de CSS, est-ce que cela a un rapport avec le fonctionnement des sites web ?

Tout à fait ! Il s'agit de **langages informatiques** qui permettent de créer des sites web. Tous les sites web sont basés sur ces langages, ils sont incontournables et universels aujourd'hui. Ils sont à la base même du Web. Le langage HTML a été inventé par un certain Tim Berners-Lee en 1991...

Tim Berners-Lee suit encore aujourd'hui avec attention l'évolution du Web. Il a créé le *World Wide Web Consortium* (W3C) qui définit les nouvelles versions des langages liés au Web. Il a par ailleurs créé plus récemment la *World Wide Web Foundation* qui analyse et suit l'évolution du Web.



De nombreuses personnes confondent (à tort) Internet et le Web. Il faut savoir que le Web *fait partie* d'Internet. Internet est un grand ensemble qui comprend, entre autres : le Web, les e-mails, la messagerie instantanée, etc. Tim Berners-Lee n'est donc pas l'inventeur d'Internet, c'est « seulement » l'inventeur du Web.

Les langages HTML et CSS sont à la base du fonctionnement de tous les sites web. Quand vous consultez un site avec votre navigateur, il faut savoir que, en coulisses, des rouages s'activent pour permettre au site web de s'afficher. L'ordinateur se base sur ce qu'on lui a expliqué en HTML et CSS pour savoir ce qu'il doit afficher, comme le montre la figure 1.3.

HTML et CSS sont deux « langues » qu'il faut savoir parler pour créer des sites web. C'est le navigateur web qui fera la traduction entre ces langages informatiques et ce que vous verrez s'afficher à l'écran. Vous vous demandez sûrement pourquoi il faut connaître deux langages pour créer des sites web ? Je vous réponds sans plus tarder !

HTML et CSS : deux langages pour créer un site web

Pour créer un site web, on doit donner des instructions à l'ordinateur. Il ne suffit pas simplement de taper le texte qui devra figurer dans le site (comme on le ferait dans un traitement de texte Word, par exemple), il faut aussi indiquer où placer ce texte, insérer des images, faire des liens entre les pages, etc.

Les rôles de HTML et CSS

Pour expliquer à l'ordinateur ce que vous voulez faire, il va falloir utiliser un langage qu'il comprend. Et c'est là que les choses se corsent, parce qu'*il va falloir apprendre deux langages* !



FIGURE 1.3 – Du HTML à l'écran



Pourquoi avoir créé deux langages ? Un seul aurait suffi, non ?

Vous devez vous dire que manipuler deux langages va être deux fois plus complexe et deux fois plus long à apprendre... mais ce n'est pas le cas ! Je vous rassure, s'il y a deux langages c'est, au contraire, pour faciliter les choses. Nous allons avoir affaire à deux langages qui *se complètent* car ils ont des rôles différents :

- **HTML** (*HyperText Markup Language*) : il a fait son apparition dès 1991 lors du lancement du Web. Son rôle est de gérer et organiser le contenu. C'est donc en HTML que vous écrirez ce qui doit être affiché sur la page : du texte, des liens, des images... Vous direz par exemple : « Ceci est mon titre, ceci est mon menu, voici le texte principal de la page, voici une image à afficher, etc. ».
- **CSS** (*Cascading Style Sheets*, aussi appelées *Feuilles de style*) : le rôle du CSS est de gérer l'apparence de la page web (agencement, positionnement, décoration, couleurs, taille du texte...). Ce langage est venu compléter le HTML en 1996.



Vous avez peut-être aussi entendu parler du langage XHTML. Il s'agit d'une variante du HTML qui se veut plus rigoureuse et qui est donc un peu plus délicate à manipuler. Pour faire simple, le HTML est apparu le premier en 1991. Début 2000, le W3C a lancé le XHTML en indiquant que ce serait l'avenir... mais le XHTML n'a pas percé comme on l'espérait. Retour aux sources en 2009 : le W3C abandonne le XHTML et décide de revenir au HTML pour le faire évoluer. Il y a beaucoup de confusion autour de ces langages, alors qu'ils se ressemblent beaucoup. Aucun n'est vraiment meilleur que l'autre, il s'agit de deux façons de faire différentes. Dans ce cours, nous allons travailler sur la dernière version de HTML (HTML5) qui est aujourd'hui le langage d'avenir que tout le monde est incité à utiliser.

Vous pouvez très bien créer un site web uniquement en HTML, mais celui-ci ne sera pas très beau : l'information apparaîtra « brute ». C'est pour cela que le langage CSS vient toujours le compléter.

Pour vous donner une idée, la figure 6.1 montre ce que donne la même page sans CSS puis avec le CSS.

Le HTML définit le contenu (comme vous pouvez le voir, c'est brut de décoffrage !). Le CSS permet, lui, d'arranger le contenu et de définir la présentation : couleurs, image de fond, marges, taille du texte...

Comme vous vous en doutez, le CSS a besoin d'une page HTML pour fonctionner. C'est pour cela que nous allons d'abord apprendre les bases du HTML avant de nous occuper de la décoration en CSS. Vos premières pages ne seront donc pas les plus esthétiques, mais qu'importe ! Cela ne durera pas longtemps.

HTML
(pas de CSS)



HTML + CSS



FIGURE 1.4 – Avec et sans CSS

Les différentes versions de HTML et CSS

Au fil du temps, les langages HTML et CSS ont beaucoup évolué. Dans la toute première version de HTML (HTML 1.0) il n'était même pas possible d'afficher des images !

Voici un très bref historique de ces langages pour votre culture générale.

Les versions de HTML

- **HTML 1** : c'est la toute première version créée par Tim Berners-Lee en 1991.
- **HTML 2** : la deuxième version du HTML apparaît en 1994 et prend fin en 1996 avec l'apparition du HTML 3.0. C'est cette version qui posera en fait les bases des versions suivantes du HTML. Les règles et le fonctionnement de cette version sont donnés par le W3C (tandis que la première version a été créée par un seul homme).
- **HTML 3** : apparue en 1996, cette nouvelle version du HTML rajoute de nombreuses possibilités au langage comme les tableaux, les applets, les scripts, le positionnement du texte autour des images, etc.
- **HTML 4** : il s'agit de la version la plus répandue du HTML (plus précisément, il s'agit de HTML 4.01). Elle apparaît pour la première fois en 1998 et propose l'utilisation de frames (qui découpent une page web en plusieurs parties), des tableaux plus complexes, des améliorations sur les formulaires, etc. Mais surtout, cette version permet pour la première fois d'exploiter des feuilles de style, notre fameux CSS !
- **HTML 5** : c'est LA dernière version. Encore assez peu répandue, elle fait beaucoup parler d'elle car elle apporte de nombreuses améliorations comme la possibilité

d'inclure facilement des vidéos, un meilleur agencement du contenu, de nouvelles fonctionnalités pour les formulaires, etc. C'est cette version que nous allons découvrir ensemble.

Les versions de CSS

- **CSS 1** : dès 1996, on dispose de la première version du CSS. Elle pose les bases de ce langage qui permet de présenter sa page web, comme les couleurs, les marges, les polices de caractères, etc.
- **CSS 2** : apparue en 1999 puis complétée par CSS 2.1, cette nouvelle version de CSS rajoute de nombreuses options. On peut désormais utiliser des techniques de positionnement très précises, qui nous permettent d'afficher des éléments où on le souhaite sur la page.
- **CSS 3** : c'est la dernière version, qui apporte des fonctionnalités particulièrement attendues comme les bordures arrondies, les dégradés, les ombres, etc.



Notez que HTML5 et CSS3 ne sont pas encore des versions totalement finalisées par le W3C. Cependant, même s'il peut y avoir des changements mineurs dans ces langages, je vous recommande chaudement de commencer dès aujourd'hui avec ces nouvelles versions. Leurs apports sont nombreux et valent vraiment le coup. D'ailleurs, de nombreux sites web professionnels se construisent aujourd'hui sur ces dernières versions.

L'éditeur de texte



De quel logiciel vais-je avoir besoin pour créer mon site web ? Vais-je devoir casser ma tirelire pour acheter un logiciel très complexe que je vais mettre des mois à comprendre ?

Il existe effectivement de nombreux logiciels dédiés à la création de sites web. Mais, je vous rassure, vous n'aurez pas à débourser un seul centime. Pourquoi aller chercher un logiciel payant et compliqué, alors que vous avez déjà tout ce qu'il faut chez vous ?

Eh oui, accrochez-vous bien parce qu'il suffit de... Bloc-Notes (figure 5.5) !

Incroyable mais vrai : on peut tout à fait créer un site web uniquement avec Bloc-Notes, le logiciel d'édition de texte intégré par défaut à Windows. D'ailleurs, j'avoue, c'est comme cela que j'ai commencé moi-même il y a quelques années.

Il y a cependant des logiciels plus puissants aujourd'hui et personne n'utilise vraiment Bloc-Notes. On peut classer ces logiciels de **création de site web** en deux catégories :

- Les **WYSIWYG** (*What You See Is What You Get* - Ce Que Vous Voyez Est Ce Que Vous Obtenez) : ce sont des programmes qui se veulent très faciles d'emploi, ils permettent de créer des sites web sans apprendre de langage particulier. Parmi les plus connus d'entre eux : Mozilla Composer, Microsoft Expression Web, Dreamweaver...

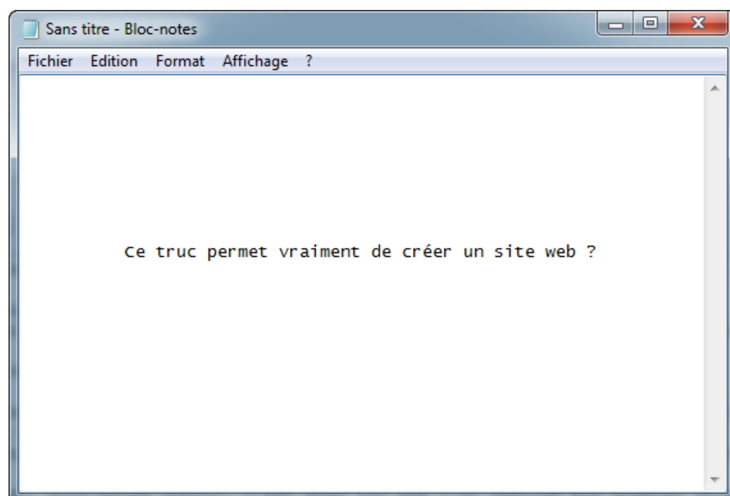


FIGURE 1.5 – Le logiciel Bloc-notes de Windows

et même Word ! Leur principal défaut est la qualité souvent assez mauvaise du code HTML et CSS qui est automatiquement généré par ces outils. Un bon créateur de site web doit tôt ou tard connaître HTML et CSS, c'est pourquoi je ne recommande pas l'usage de ces outils.
– Les **éditeurs de texte** : ce sont des programmes dédiés à l'écriture de code. On peut en général les utiliser pour de multiples langages, pas seulement HTML et CSS. Ils se révèlent être de puissants alliés pour les créateurs de sites web !

Vous l'aurez compris, je vais vous inviter à utiliser un éditeur de texte dans ce cours. Voici quelques conseils, selon que vous êtes sous Windows, Mac OS X ou Linux.

Sous Windows

Il existe une grande quantité d'éditeurs de texte, je ne pourrai pas vous les présenter tous. Néanmoins, je vous invite à vous pencher sur Notepad++, l'un des plus utilisés parmi ceux disponibles pour Windows. Ce logiciel est simple, en français et gratuit.

▷ Site web de Notepad++
Code web : 642453



Si vous ne savez pas utiliser les codes web, je vous renvoie à la page iii de ce livre, que vous trouverez dans l'avant-propos.

Voici à quoi ressemble Notepad++ lorsque vous le lancez (figure 1.6).

Je vous conseille de faire la manipulation suivante : allez dans le menu `Langage` > `H` > `HTML`. Cela permettra au logiciel de savoir que l'on va taper du HTML. Lorsque vous

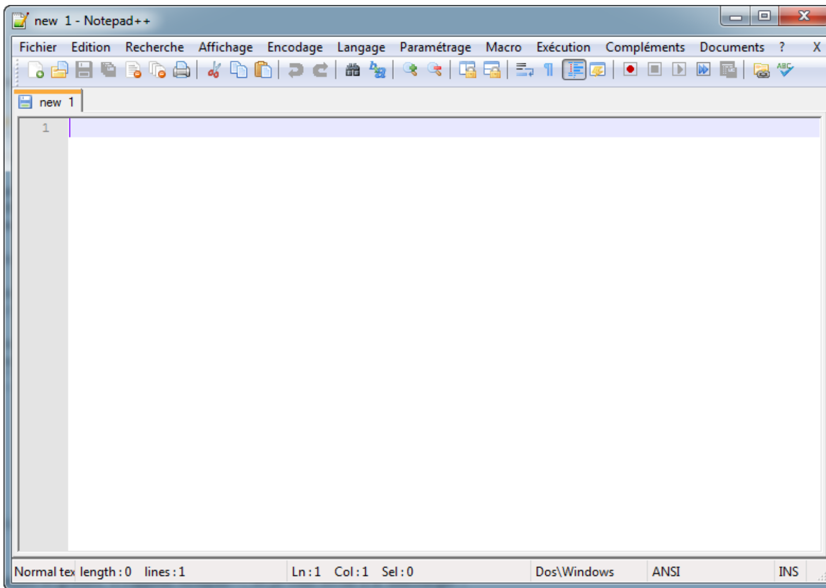


FIGURE 1.6 – Notepad++

utiliserez le logiciel, il colorera votre code (figure 1.7), ce qui vous permettra de vous repérer plus facilement.

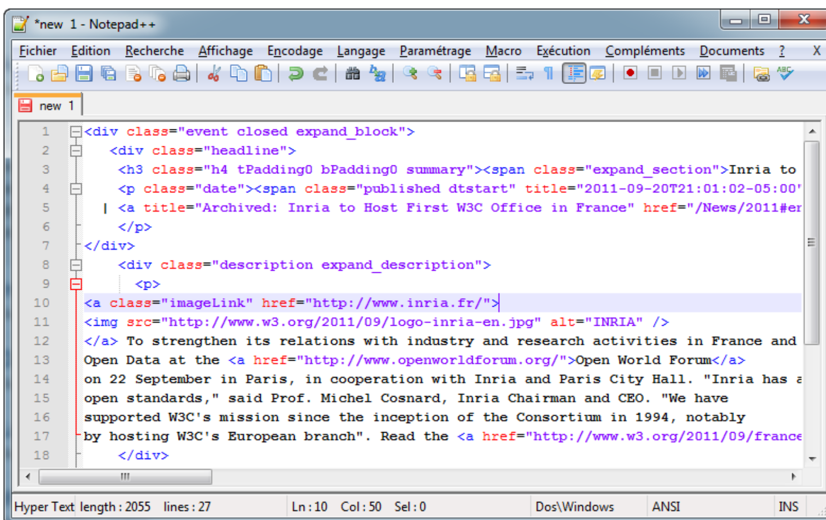


FIGURE 1.7 – Coloration sous Notepad++



Pour l'instant, ne vous préoccupez pas de savoir ce que signifie tout ce charabia que vous pouvez voir. Je souhaitais simplement vous donner un aperçu des possibilités du logiciel.

Il existe d'autres éditeurs disponibles sous Windows. Si Notepad++ ne vous convient pas, vous pouvez essayer :

- jEdit ;
- PSpad ;
- ConTEXT ;
- ... et bien d'autres si vous recherchez « Éditeur de texte » sur le Web.

Sous Mac OS X

Vous pouvez essayer l'un des logiciels suivants :

- jEdit ;
- Smultron ;
- TextWrangler.

Sous Linux

Les éditeurs de texte sont légion sous Linux. Certains d'entre eux sont installés par défaut, d'autres peuvent être téléchargés facilement *via* le centre de téléchargement (sous Ubuntu notamment) ou au moyen de commandes comme `apt-get` et `aptitude`. Voici quelques logiciels que vous pouvez tester :

- gEdit ;
- Kate ;
- vim ;
- Emacs ;
- jEdit.

Les navigateurs

Pourquoi le navigateur est important

Le navigateur est le programme qui nous permet de voir les sites web. Comme je vous l'ai expliqué plus tôt, le travail du navigateur est de lire le code HTML et CSS pour afficher un résultat visuel à l'écran. Si votre code CSS dit « Les titres sont en rouge », alors le navigateur affichera les titres en rouge. Le rôle du navigateur est donc essentiel !

On ne dirait pas, mais un navigateur est un programme extrêmement complexe. En effet, comprendre le code HTML et CSS n'est pas une mince affaire. Le principal problème, vous vous en rendrez vite compte, c'est que *les différents navigateurs n'affichent*

pas le même site exactement de la même façon ! Il faudra vous y faire et prendre l'habitude de vérifier régulièrement que votre site fonctionne correctement sur la plupart des navigateurs.

Les navigateurs sur ordinateur

Télécharger les navigateurs

Il existe de nombreux navigateurs différents. Voici les principaux à connaître :



Google Chrome : le navigateur de Google, simple d'emploi et très rapide.
C'est le navigateur que j'utilise au quotidien.
Compatibilité : Windows, Mac et Linux
Code web : 901626



Mozilla Firefox : le navigateur de la fondation Mozilla, célèbre et réputé. Je l'utilise fréquemment pour tester mes sites web.
Compatibilité : Windows, Mac et Linux
Code web : 882142



Internet Explorer : le navigateur de Microsoft, qui équipe tous les PC Windows. Je l'utilise fréquemment pour tester mes sites web.
Compatibilité : Windows
Code web : 577078



Safari : le navigateur d'Apple, qui équipe tous les Mac.
Compatibilité : Windows et Mac
Code web : 521278



Opera : l'éternel *outsider*. Il est moins utilisé mais propose de nombreuses fonctionnalités.
Compatibilité : Windows, Mac et Linux
Code web : 260892



Il est conseillé d'installer plusieurs navigateurs sur son ordinateur pour s'assurer que son site fonctionne correctement sur chacun d'eux. De manière générale, je conseille de tester son site web régulièrement au moins sur Google Chrome, Mozilla Firefox et Internet Explorer. Notez que Safari et Google Chrome affichent les sites web quasiment de la même façon. Il n'est pas forcément nécessaire de tester son site sur Safari et Google Chrome, même si c'est toujours plus sûr.

La figure 1.8 vous montre un aperçu du résultat produit par quelques-uns de ces prin-

cipaux navigateurs sur la page d'accueil de Google.



FIGURE 1.8 – Aperçu de quelques navigateurs

Comprendre les différences entre navigateurs

Comme je vous le disais plus tôt, les navigateurs n'affichent pas toujours un même site web *exactement* de la même façon. Pourquoi ? Cela est dû au fait que les navigateurs ne connaissent pas toujours les dernières fonctionnalités de HTML et CSS. Par exemple, Internet Explorer a longtemps été en retard sur certaines fonctionnalités CSS (et paradoxalement, il a aussi été en avance sur quelques autres).

Pour compliquer les choses, plusieurs versions des navigateurs co-existent :

- Firefox 2, Firefox 3.5, Firefox 3.6, Firefox 4 ;
- Internet Explorer 6, Internet Explorer 7, Internet Explorer 8, Internet Explorer 9 ;
- Chrome 8, Chrome 9, Chrome 10 ;
- etc.

Chaque version prend en charge de nouvelles fonctionnalités mais, si les utilisateurs ne mettent pas à jour leur(s) navigateur(s), cela devient un problème pour les **webmasters** comme vous qui créent des sites web. Chrome a résolu en grande partie le problème en mettant en place des mises à jour automatiques, sans intervention de l'utilisateur. Les utilisateurs de Firefox ne pensent pas toujours à mettre à niveau leur navigateur ; quant à Internet Explorer, les utilisateurs sont d'autant moins incités à mettre à jour leur navigateur que les dernières versions nécessitent aussi de passer à une version récente de Windows (Internet Explorer 9 n'est pas disponible pour Windows XP, par exemple).

Des sites comme **normansblog.de** et **caniuse.com** tiennent notamment à jour une liste des fonctionnalités CSS prises en charge par les différentes versions de chaque navigateur (figure 1.9).

▷ Liste sur *normansblog.de*
Code web : 720397

Comme vous le voyez, c'est... compliqué.

Browser Rendering Engine	Firefox Gecko					Safari Webkit			Chrome Webkit		Internet Explorer Trident			Opera Presto	
Version	3.6	4	5	6	7b – 8a	4	5	5.1	9	10 – 14	6 – 8	9	10a3	10.6 – 11	11.1 – 11.5
Animations	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✗
Background Gradients	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	⚠
Background Size	✓	✓	✓	✓	✓	⚠	✓	✓	✓	✓	✗	✓	✓	✓	✓
Border Image	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
Border Radius	⚠	⚠	⚠	⚠	⚠	⚠	✓	✓	✓	✓	✗	✓	✓	✓	✓
Box Shadow	✓	✓	✓	✓	✓	⚠	✓	✓	⚠	✓	✗	✓	✓	✓	✓
Columns	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗	✓
Font Face	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	⚠	✓	✓	✓	✓
HSLa	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Hyphens	✗	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
Multiple Backgrounds	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Opacity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓

FIGURE 1.9 – Les fonctionnalités gérées par les différents navigateurs

L'essentiel des soucis viendra le plus souvent des anciennes versions d'Internet Explorer (IE6, IE7, IE8). Il faudra vérifier comment le site s'affiche sous ces anciennes versions... Attendez-vous à des surprises ! Vérifiez surtout que votre site s'affiche sans erreur, sans chercher à obtenir exactement le même rendu sur les vieilles versions de ces navigateurs.



Il existe sous Windows un programme appelé IETester. Il permet de vérifier le rendu son site sous différentes versions d'Internet Explorer. À noter que ce programme est relativement instable (il plante souvent) mais il a le mérite d'exister.

▷ Télécharger IETester
Code web : 541385

Les navigateurs sur mobile

En plus des navigateurs que je vous ai présentés, il faut savoir qu'il existe des variantes de ces navigateurs conçues pour les téléphones portables, en particulier pour les **smartphones**. De plus en plus de personnes consultent aujourd'hui des sites web sur leur portable, il faut donc connaître un minimum le fonctionnement des navigateurs des téléphones.

En fait, vous n'allez pas être dépaysés : la plupart des navigateurs sur smartphones sont les mêmes que sur ordinateur, dans une version plus légère adaptée aux mobiles. Tout dépend du type de téléphone.

- **iPhone** : sur l'iPhone d'Apple, le navigateur utilisé est Safari Mobile. Il s'agit d'une version *light* et néanmoins très complète de Safari pour ordinateur.
- **Android** : les portables sous Android bénéficient du navigateur Chrome Mobile. Là encore, il s'agit d'une version adaptée aux mobiles.

- **Windows Phone** : sous Windows Phone, on retrouve... Internet Explorer Mobile ! Le principe est le même que pour les précédents navigateurs : il s'agit d'une version dédiée aux mobiles.
- **Blackberry** : les Blackberry font exception car ils ont leur propre navigateur (il n'existe pas d'équivalent sur ordinateur). Néanmoins, les versions les plus récentes de ce navigateur se basent sur un noyau commun à Safari et Chrome (il s'agit du moteur de rendu Webkit). Par conséquent, l'affichage est en général proche de celui proposé par Safari et Chrome.

Les navigateurs pour mobiles prennent en charge la plupart des dernières fonctionnalités de HTML et CSS. De plus, le système de mise à jour automatisé des mobiles nous garantit que les utilisateurs auront le plus souvent les dernières versions.

Sachez néanmoins que des différences existent entre ces différents navigateurs mobiles et qu'il est conseillé de tester son site sur ces appareils aussi ! En particulier, l'écran étant beaucoup moins large, il faudra vérifier que votre site s'affiche correctement.



Les tablettes tactiles sont équipées des mêmes navigateurs, l'écran est simplement plus large. Ainsi, l'iPad est fourni avec Safari Mobile.

En résumé

- Le Web a été inventé par Tim Berners-Lee au début des années 1990.
- Pour créer des sites web, on utilise deux langages informatiques :
  - HTML : permet d'écrire et organiser le contenu de la page (paragraphes, titres...) ;
  - CSS : permet de mettre en forme la page (couleur, taille...).
- Il y a eu plusieurs versions des langages HTML et CSS. Les dernières versions sont HTML5 et CSS3.
- Le navigateur web est un programme qui permet d'afficher des sites web. Il lit les langages HTML et CSS pour savoir ce qu'il doit afficher.
- Il existe de nombreux navigateurs web différents : Google Chrome, Mozilla Firefox, Internet Explorer, Safari, Opera... Chacun affiche un site web de manière légèrement différente des autres navigateurs.
- Dans ce cours, nous allons apprendre à utiliser les langages HTML et CSS. Nous travaillerons dans un programme appelé « éditeur de texte » (Notepad++, jEdit, vim...).

Chapitre 2

Votre première page web en HTML

Difficulté :

Ça y est, vous avez installé tous les logiciels ? Vous devriez maintenant avoir un éditeur de texte pour *créer votre site* (comme Notepad++) et plusieurs navigateurs pour le *tester* (Mozilla Firefox, Google Chrome, Internet Explorer. . .).

Dans ce chapitre, nous allons commencer à pratiquer ! Nous allons découvrir les bases du langage HTML et enregistrer notre toute première page web ! Alors oui, bien sûr, ne vous attendez pas encore à réaliser une page web exceptionnelle dès ce second chapitre, mais patience. . . cela viendra !



Créer une page web avec l'éditeur

Allez, mettons-nous en situation ! Comme je vous l'ai dit, nous allons créer notre site dans un éditeur de texte. Vous avez dû en installer un suite à mes conseils dans le premier chapitre : qu'il s'appelle Notepad++, PSpad, jEdit, vim, TextWrangler... peu importe. Ces logiciels ont un but très simple : vous permettre d'écrire du texte !

Dans la suite de ce cours, je travaillerai avec Notepad++. Je vais donc l'ouvrir (figure 2.1).

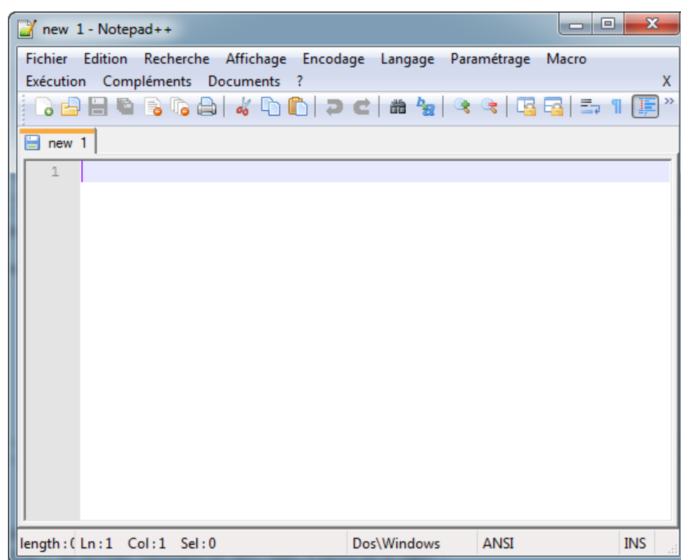


FIGURE 2.1 – Ouverture de Notepad++

Bon, qu'est-ce qu'on fait maintenant ? Qu'est-ce qu'on écrit sur cette feuille blanche ?

On va faire un petit essai. Je vous invite à écrire ce qui vous passe par la tête, comme moi à la figure 2.2.

Vous pouvez écrire les mêmes phrases que moi ou ce que vous voulez ; le but est d'écrire quelque chose.

Maintenant, enregistrons ce fichier. Pour cela, c'est très simple : comme dans tous les programmes, vous avez un menu `Fichier` > `Enregistrer`. Une boîte de dialogue vous demande où enregistrer le fichier et sous quel nom. Enregistrez-le où vous voulez. Donnez au fichier le nom que vous voulez, en terminant par `.html`, par exemple `test.html`, comme indiqué à la figure 2.3.



Je vous recommande de créer un nouveau dossier dans vos documents qui contiendra les fichiers de votre site. Pour ma part j'ai créé un dossier `test` dans lequel j'ai mis mon fichier `test.html`.

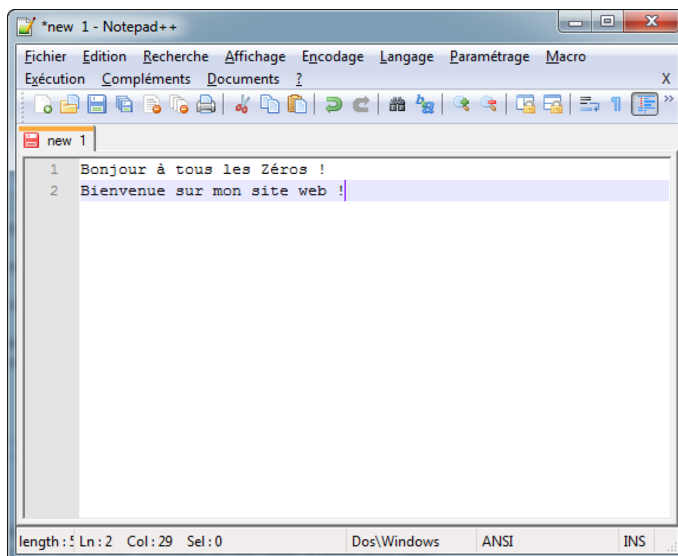


FIGURE 2.2 – Du texte dans Notepad++

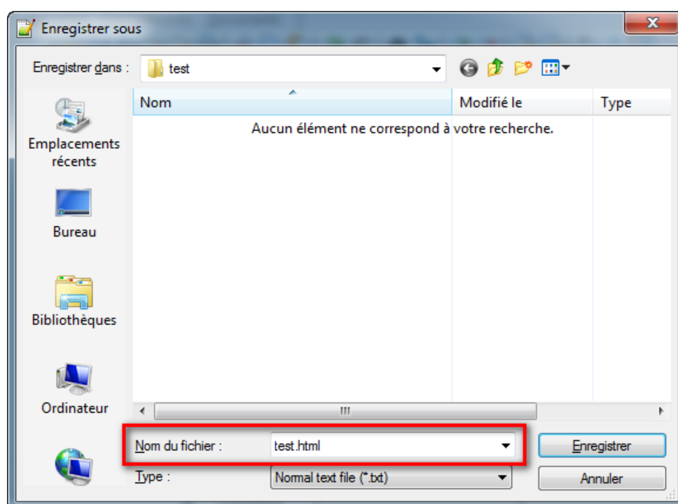


FIGURE 2.3 – Enregistrement d'un fichier sous Notepad++

Ouvrez maintenant l'explorateur de fichiers dans le dossier où vous avez enregistré votre page. Vous y verrez le fichier que vous venez de créer (figure 2.4).

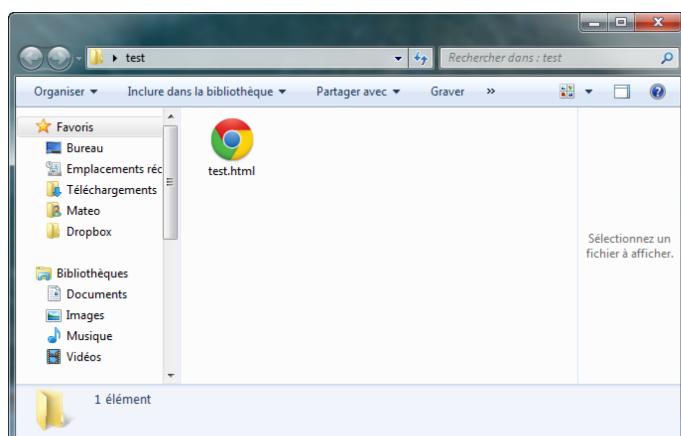


FIGURE 2.4 – Le fichier dans l'explorateur

L'icône qui représente le fichier dépend de votre navigateur web par défaut. Ici, l'icône est celle de Google Chrome, mon navigateur par défaut, mais le fichier a peut-être une autre icône chez vous. Voici par exemple les icônes qui apparaissent lorsque votre navigateur principal est Firefox ou Internet Explorer (figure 2.5).

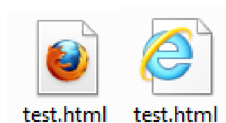


FIGURE 2.5 – Icône d'un fichier HTML sous Firefox et Internet Explorer

Faites simplement un double-clic sur ce fichier et. . . votre navigateur s'ouvre et, comme à la figure 2.6, affiche le texte que vous avez écrit.



Cela ne marche pas bien, on dirait ! Tout le texte s'affiche sur la même ligne alors qu'on avait écrit deux lignes de texte différentes ! ?

En effet, bien vu ! Le texte s'affiche sur la même ligne alors qu'on avait demandé à l'écrire sur deux lignes différentes. Que se passe-t-il ?

En fait, pour créer une page web il ne suffit pas de taper simplement du texte comme on vient de le faire. En plus de ce texte, il faut aussi écrire ce qu'on appelle des **balises**, qui vont donner des instructions à l'ordinateur comme « aller à la ligne », « afficher une image », etc.

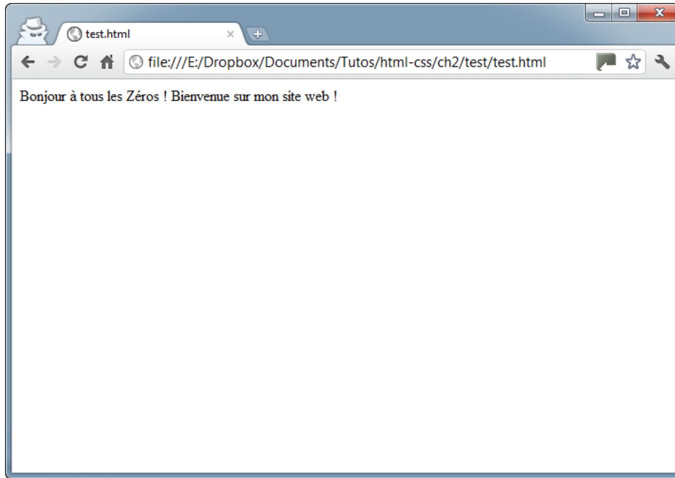


FIGURE 2.6 – La page web affichée

Les balises et leurs attributs

Bon, tout cela était trop facile. Évidemment, il a fallu que ces satanés informaticiens s'en mêlent et compliquent les choses. Il ne suffit pas d'écrire « simplement » du texte dans l'éditeur, il faut aussi donner des instructions à l'ordinateur. En HTML, on utilise pour cela des **balises**.

Les balises

Les pages HTML sont remplies de ce qu'on appelle des balises. Celles-ci sont invisibles à l'écran pour vos visiteurs, mais elles permettent à l'ordinateur de comprendre ce qu'il doit afficher. Les balises se repèrent facilement. Elles sont entourées de « chevrons », c'est-à-dire des symboles < et >, comme ceci : `<balise>`

À quoi est-ce qu'elles servent ? Elles indiquent la nature du texte qu'elles encadrent. Elles veulent dire par exemple : « Ceci est le titre de la page », « Ceci est une image », « Ceci est un paragraphe de texte », etc.

On distingue deux types de balises : les balises en paires et les balises orphelines.

Les balises en paires

Elles s'ouvrent, contiennent du texte, et se ferment plus loin. Voici à quoi elles ressemblent :

```
<titre>Ceci est un titre</titre>
```

On distingue une balise ouvrante (<titre>) et une balise fermante (</titre>) qui indique que le titre se termine. Cela signifie pour l'ordinateur que tout ce qui n'est *pas*

entre ces deux balises... n'est pas un titre.

```
Ceci n'est pas un titre <titre>Ceci est un titre</titre> Ceci n
    'est pas un titre
```

Les balises orphelines

Ce sont des balises qui servent le plus souvent à insérer un élément à un endroit précis (par exemple une image). Il n'est pas nécessaire de délimiter le début et la fin de l'image, on veut juste dire à l'ordinateur « Insère une image ici ».

Une balise orpheline s'écrit comme ceci :

```
<image />
```



Notez que le / de fin n'est pas obligatoire. On pourrait écrire seulement <image>. Néanmoins, pour ne pas les confondre avec le premier type de balise, les webmasters recommandent de rajouter ce / (*slash*) à la fin des balises orphelines. Vous me verrez donc mettre un / aux balises orphelines et je vous recommande de faire de même, c'est une bonne pratique.

Les attributs

Les attributs sont un peu les options des balises. Ils viennent les compléter pour donner des informations supplémentaires. L'attribut se place après le nom de la balise ouvrante et a le plus souvent une valeur, comme ceci :

```
<balise attribut="valeur">
```

À quoi cela sert-il ? Prenons la balise <image /> que nous venons de voir. Seule, elle ne sert pas à grand chose. On pourrait rajouter un attribut qui indique le nom de l'image à afficher :

```
<image nom="photo.jpg" />
```

L'ordinateur comprend alors qu'il doit afficher l'image contenue dans le fichier `photo.jpg`.

Dans le cas d'une balise fonctionnant « par paire », on ne met les attributs que dans la balise ouvrante et pas dans la balise fermante. Par exemple, ce code indique que la citation est de Neil Armstrong et qu'elle date du 21 Juillet 1969 :

```
<citation auteur="Neil Armstrong" date="21/07/1969">
C'est un petit pas pour l'homme, mais un bond de géant pour l'
    humanité.
</citation>
```



Toutes les balises que nous venons de voir sont fictives. Les vraies balises ont des noms en anglais (eh oui !), nous allons les découvrir dans la suite de ce cours.

Structure de base d'une page HTML5

Reprenons notre éditeur de texte (dans mon cas Notepad++). Je vous invite à écrire ou à copier-coller le code source ci-dessous dans Notepad++. Ce code correspond à la base d'une page web en HTML5 :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titre</title>
    </head>

    <body>

    </body>
</html>
```

▷ Copier ce code
Code web : 310363



J'ai mis des espaces au début de certaines lignes pour « décaler » les balises. Ce n'est pas obligatoire et cela n'a aucun impact sur l'affichage de la page, mais cela rend le code source plus lisible. On appelle cela l'**indentation**. Dans votre éditeur, il suffit d'appuyer sur la touche Tab pour avoir le même résultat.

Copié dans Notepad++, cela donne la figure 2.7.

Vous noterez que les balises s'ouvrent et se ferment dans un ordre précis. Par exemple, la balise `<html>` est la première que l'on ouvre et c'est aussi la dernière que l'on ferme (tout à la fin du code, avec `</html>`). *Les balises doivent être fermées dans le sens inverse de leur ouverture.* Un exemple :

- `<html><body></body></html>` : **correct**. Une balise qui est ouverte à l'intérieur d'une autre doit aussi être fermée à l'intérieur.
- `<html><body></html></body>` : **incorrect**, les balises s'entremêlent.



Euh, on pourrait avoir des explications sur toutes les balises que l'on vient de copier dans l'éditeur, m'sieur ?

Bien sûr, c'est demandé si gentiment. Ne prenez pas peur en voyant toutes ces balises d'un coup, je vais vous expliquer leur rôle !

Le doctype

```
<!DOCTYPE html>
```

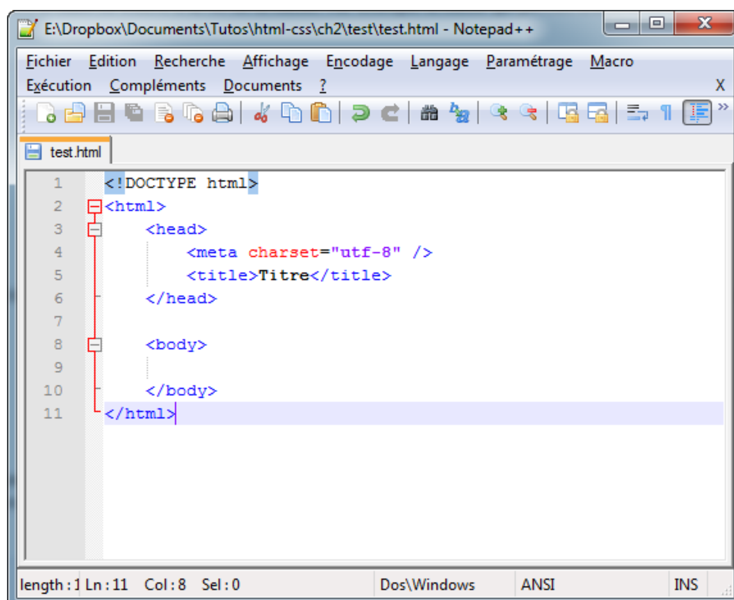


FIGURE 2.7 – Code HTML5 minimal dans Notepad++

La toute première ligne s'appelle le **doctype**. Elle est indispensable car c'est elle qui indique qu'il s'agit bien d'une page web HTML. Ce n'est pas vraiment une balise comme les autres (elle commence par un point d'exclamation), vous pouvez considérer que c'est un peu l'exception qui confirme la règle.



Cette ligne du doctype était autrefois incroyablement complexe. Il était impossible de la retenir de tête. Pour XHTML 1.0, il fallait écrire : `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">`. Dans le cadre de HTML5, il a été décidé de la simplifier, pour le plus grand bonheur des webmasters. Quand vous voyez une balise doctype courte (`<!DOCTYPE html>`), cela signifie que la page est écrite en HTML5.

La balise </html>

```
<html>

</html>
```

C'est la balise principale du code. Elle englobe tout le contenu de votre page. Comme vous pouvez le voir, la balise fermante `</html>` se trouve tout à la fin du code !

L'en-tête <head> et le corps <body>

Une page web est constituée de deux parties :

- L'en-tête <head> : cette section donne quelques informations générales sur la page comme son titre, l'encodage (pour la gestion des caractères spéciaux), etc. Cette section est généralement assez courte. Les informations que contient l'en-tête ne sont pas affichées sur la page, ce sont simplement des informations générales à destination de l'ordinateur. Elles sont cependant très importantes !
- Le corps <body> : c'est là que se trouve la partie principale de la page. Tout ce que nous écrirons ici sera affiché à l'écran. C'est à l'intérieur du corps que nous écrirons la majeure partie de notre code.

Pour le moment, le corps est vide (nous y reviendrons plus loin). Intéressons-nous par contre aux deux balises contenues dans l'en-tête...

L'encodage (charset)

```
<meta charset="utf-8" />
```

Cette balise indique l'encodage utilisé dans votre fichier `.html`.

Sans rentrer dans les détails, car cela pourrait vite devenir compliqué, l'encodage indique la façon dont le fichier est enregistré. C'est lui qui détermine comment les caractères spéciaux vont s'afficher (accents, idéogrammes chinois et japonais, caractères arabes, etc.).

Il y a plusieurs techniques d'encodage portant des noms bizarres et utilisées en fonction des langues : ISO-8859-1, OEM 775, Windows-1253... Une seule cependant devrait être utilisée aujourd'hui autant que possible : UTF-8. Cette méthode d'encodage permet d'afficher sans aucun problème pratiquement tous les symboles de toutes les langues de notre planète ! C'est pour cela que j'ai indiqué `utf-8` dans cette balise.

Il faut aussi que votre fichier soit bien enregistré en UTF-8. C'est le cas le plus souvent sous Linux par défaut mais, sous Windows, il faut généralement le dire au logiciel.



Sous Notepad++, allez dans le menu `Encodage > Encoder en UTF-8 (sans BOM)` pour que votre fichier soit enregistré en UTF-8 dès le début. Cela ne s'applique qu'au fichier actuellement ouvert. Pour ne pas avoir à le faire pour chaque nouveau fichier, je vous conseille d'aller dans le menu `Paramétrage > Préférences`, onglet `Nouveau document/Dossier`. Sélectionnez `UTF-8 sans BOM` dans la liste.



Si vous avez un problème d'affichage des accents plus tard sur votre page web, c'est qu'il y a un problème avec l'encodage. Vérifiez que la balise indique bien UTF-8 et que votre fichier est enregistré en UTF-8 (votre éditeur de texte est capable de vous le dire, Notepad++ le fait dans le menu `Encodage`).

Le titre principal de la page

```
<title>
```

C'est le titre de votre page, probablement l'élément le plus important ! Toute page doit avoir un titre qui décrit ce qu'elle contient. Il est conseillé de garder le titre assez court (moins de 100 caractères en général).

Le titre ne s'affiche pas dans votre page mais en haut de celle-ci (souvent dans l'onglet du navigateur). Enregistrez votre page web et ouvrez-la dans votre navigateur. Vous verrez que le titre s'affiche dans l'onglet, comme sur la figure 2.8.

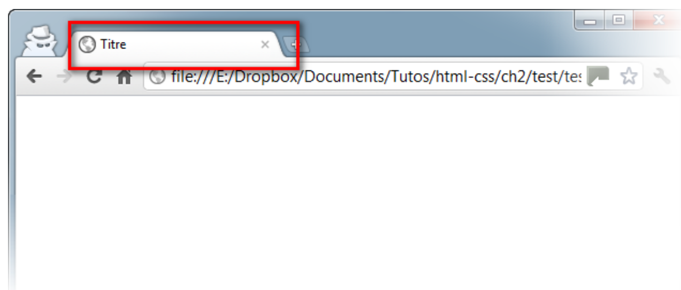


FIGURE 2.8 – Le titre de la page apparaît en haut du navigateur

Il faut savoir que le titre apparaît aussi dans les résultats de recherche, comme sur Google (figure 2.9).

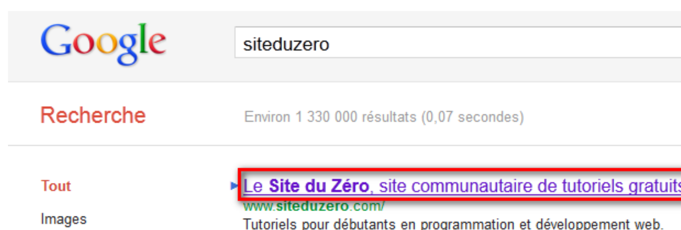


FIGURE 2.9 – Le titre de la page apparaît dans les recherches Google

Autant vous dire que bien choisir son titre est important !

Les commentaires

Nous avons appris à créer notre première *vraie* page HTML dans ce chapitre. Avant de terminer, j'aimerais vous présenter le principe des commentaires.

Un **commentaire** en HTML est un texte qui sert simplement de mémo. Il n'est pas affiché, il n'est pas lu par l'ordinateur, cela ne change rien à l'affichage de la page.



Bref, cela ne sert à rien ?

Eh bien si ! Cela sert à *vous* et aux personnes qui liront le code source de votre page. Vous pouvez utiliser les commentaires pour laisser des indications sur le fonctionnement de votre page.

Quel intérêt ? Cela vous permettra de vous rappeler comment fonctionne votre page si vous revenez sur votre code source après un long moment d'absence. Ne rigolez pas, cela arrive à tous les webmasters.

Insérer un commentaire

Un commentaire est une balise HTML avec une forme bien spéciale :

```
<!-- Ceci est un commentaire -->
```

Vous pouvez le mettre où vous voulez au sein de votre code source : il n'a aucun impact sur votre page, mais vous pouvez vous en servir pour vous aider à vous repérer dans votre code source (surtout s'il est long).

```
<!DOCTYPE html>
<html>
    <head>
        <!-- En-tête de la page -->
        <meta charset="utf-8" />
        <title>Titre</title>
    </head>

    <body>
        <!-- Corps de la page -->
    </body>
</html>
```

Tout le monde peut voir vos commentaires. . . et tout votre code HTML !

Terminons par une remarque importante : *tout le monde peut voir le code HTML de votre page* une fois celle-ci mise en ligne sur le Web. Il suffit de faire un clic droit sur la page et de sélectionner « Afficher le code source de la page » (l'intitulé peut changer selon votre navigateur), comme le montre la figure 2.10.

Le code source s'affiche alors (figure 2.11).

Vous pouvez tester cette manipulation sur n'importe quel site web, cela marche ! Garanti à 100%. Cela s'explique assez facilement : le navigateur *doit* obtenir le code HTML pour savoir ce qu'il faut afficher. Le code HTML de tous les sites est donc public.

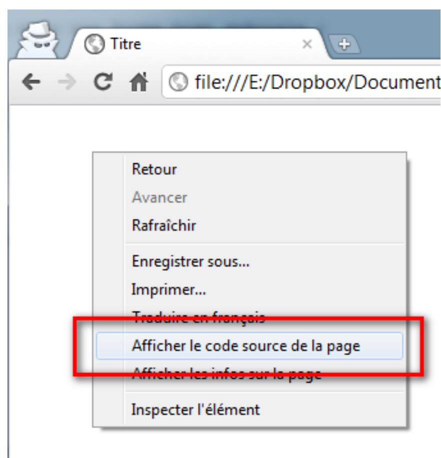


FIGURE 2.10 – Menu afficher le code source

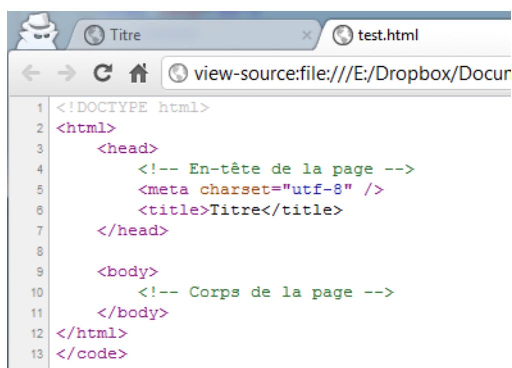


FIGURE 2.11 – Affichage du code source

La morale de l'histoire ? Tout le monde pourra voir votre code HTML et vous ne pouvez pas l'empêcher. Par conséquent, ne mettez pas d'informations sensibles comme des mots de passe dans les commentaires... et soignez votre code source, car je pourrai venir vérifier si vous avez bien suivi mon cours à la lettre !



Lorsque vous regarderez le code de certains sites web, ne prenez pas peur s'il vous paraît long ou ne pas respecter les mêmes règles que celles que je vous présente dans ce livre. Tous les sites ne sont pas écrits en HTML5 (loin de là) et, parfois, certains webmasters rédigent très mal leur code, ce ne sont pas toujours des exemples à suivre !

En résumé

- On utilise l'éditeur de texte (Notepad++, jEdit, vim...) pour créer un fichier ayant l'extension `.html` (par exemple : `test.html`). Ce sera notre page web.
- Ce fichier peut être ouvert dans le navigateur web simplement en faisant un double-clic dessus.
- À l'intérieur du fichier, nous écrirons le contenu de notre page, accompagné de balises HTML.
- Les balises peuvent avoir plusieurs formes :
  - `<balise> </balise>` : elles s'ouvrent et se ferment pour délimiter le contenu (début et fin d'un titre, par exemple).
  - `<balise />` : balises orphelines (on ne les insère qu'en un seul exemplaire), elles permettent d'insérer un élément à un endroit précis (par exemple une image).
- Les balises sont parfois accompagnées d'attributs pour donner des indications supplémentaires (exemple : `<image nom="photo.jpg" />`).
- Une page web est constituée de deux sections principales : un en-tête (`<head>`) et un corps (`<body>`).
- On peut afficher le code source de n'importe quelle page web en faisant un clic droit puis en sélectionnant `Afficher le code source de la page`.

Chapitre 3

Organiser son texte

Difficulté :

Bon , la page blanche c'est bien joli, mais votre site web risque d'avoir un succès mitigé si vous le laissez comme cela.

Nous allons découvrir de nombreuses balises HTML dans ce chapitre. Certaines existent depuis la toute première version de HTML, d'autres ont été introduites plus récemment dans HTML5.

Nous allons voir successivement dans ce chapitre :

- comment rédiger des paragraphes ;
- comment structurer sa page avec les titres ;
- comment donner de l'importance à certains mots de son texte ;
- comment organiser les informations sous forme de listes.

Motivés ? Allez, vous allez voir, ce n'est pas compliqué.



Les paragraphes

La plupart du temps, lorsqu'on écrit du texte dans une page web, on le fait à l'intérieur de paragraphes. Le langage HTML propose justement la balise <p> pour délimiter les paragraphes.

```
<p>Bonjour et bienvenue sur mon site !</p>
```

- <p> signifie « Début du paragraphe » ;
- </p> signifie « Fin du paragraphe ».

Comme je vous l'ai dit au chapitre précédent, on écrit le contenu du site web entre les balises <body></body>. Il nous suffit donc de mettre notre paragraphe entre ces deux balises et nous aurons enfin notre première vraie page web avec du texte !

Je reprends donc exactement le même code qu'au chapitre précédent et j'y ajoute mon paragraphe :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Paragraphes</title>
    </head>

    <body>
        <p>Bonjour et bienvenue sur mon site !</p>
    </body>
</html>
```

Essayez, vous allez voir le résultat ! Bon, ok, ce n'est pas encore le nirvana mais c'est un bon début.

Mais ne nous arrêtons pas en si bon chemin. Nous allons voir maintenant quelque chose d'un peu particulier en HTML : le saut de ligne. Cela paraît simple et pourtant, cela ne fonctionne pas vraiment comme dans un traitement de texte habituel...

Sauter une ligne

En HTML, si vous appuyez sur la touche Entrée, cela ne crée pas une nouvelle ligne comme vous en avez l'habitude. Essayez donc ce code :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Essais de sauts de ligne</title>
    </head>

    <body>
        <p>Bonjour et bienvenue sur mon site !
```

```
                Ceci est mon premier test alors soyez indulgents s'
                    il vous plaît, j'apprends petit à petit comment
                    cela marche.
                Pour l'instant c'est un peu vide, mais revenez dans
                     2-3 jours quand j'aurai appris un peu plus de
                    choses, je vous assure que vous allez être
                    surpris !</p>
        </body>
</html>
```

Tout le texte s'affiche sur la même ligne alors qu'on est bien allé à la ligne dans le code ! Taper frénétiquement sur la touche `Entrée` dans l'éditeur de texte ne sert donc strictement à rien.

Comme vous devez vous en douter, il y a pourtant bien un moyen de faire des sauts de ligne en HTML.

En fait, si vous voulez écrire un deuxième paragraphe, il vous suffit d'utiliser une deuxième balise <p>. Votre code HTML devrait donc être au final rempli de balises de paragraphe !

Un exemple :

```
<html>
    <head>
        <meta charset="utf-8" />
        <title>Paragraphes</title>
    </head>

    <body>
        <p>Bonjour et bienvenue sur mon site !
        Ceci est mon premier test alors soyez indulgents s'il
            vous plaît, j'apprends petit à petit comment cela
            marche.</p>

        <p>Pour l'instant c'est un peu vide, mais revenez dans
            2-3 jours quand j'aurai appris un peu plus de choses
            , je vous assure que vous allez être surpris !</p>
    </body>
</html>
```

Le résultat se trouve à la figure 3.1.



Oui, mais si je veux juste aller à la ligne dans un paragraphe et non pas sauter une ligne ?

Eh bien devinez quoi : il existe une balise « Aller à la ligne » ! C'est une balise **orpheline** qui sert juste à indiquer qu'on doit aller à la ligne : `<br />`. Vous devez obligatoirement la mettre *à l'intérieur d'un paragraphe*.

Voici comment l'utiliser dans un code :

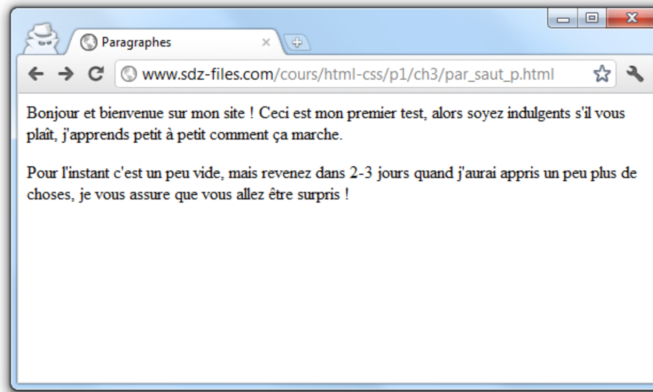


FIGURE 3.1 – Deux paragraphes avec 2 balises <p>

```
<html>
    <head>
        <meta charset="utf-8" />
        <title>Sauts de ligne</title>
    </head>

    <body>
        <p>Bonjour et bienvenue sur mon site !<br />
        Ceci est mon premier test alors soyez indulgents s'il vous plaît, j'apprends petit à petit comment cela marche.</p>

        <p>Pour l'instant c'est un peu vide, mais revenez dans 2-3 jours quand j'aurai appris un peu plus de choses, je vous assure que vous allez être surpris !</p>
    </body>
</html>
```



Vous pouvez théoriquement mettre plusieurs balises
 d'affilée pour faire plusieurs sauts de lignes, mais on considère que c'est une mauvaise pratique qui rend le code délicat à maintenir. Pour décaler un texte avec plus de précision, on utilisera le CSS, ce langage qui vient compléter le HTML et dont je vous parlerai un peu plus loin.

Donc c'est compris ?

- <p> </p> : pour organiser son texte en paragraphes ;
-
 : pour aller à la ligne.

Maintenant qu'on sait écrire des paragraphes, voyons voir comment on crée des titres.

Les titres

Lorsque le contenu de votre page va s'étoffer avec de nombreux paragraphes, il va devenir difficile pour vos visiteurs de se repérer. C'est là que les titres deviennent utiles.

En HTML, on est verni, on a le droit d'utiliser six niveaux de titres différents. Je veux dire par là qu'on peut dire « Ceci est un titre très important », « Ceci est un titre un peu moins important », « Ceci est un titre encore moins important », etc. On a donc six balises de titres différentes :

- `<h1> </h1>` : signifie « titre très important ». En général, on s'en sert pour afficher le titre de la page au début de celle-ci.
- `<h2> </h2>` : signifie « titre important ».
- `<h3> </h3>` : pareil, c'est un titre un peu moins important (on peut dire un « sous-titre » si vous voulez).
- `<h4> </h4>` : titre encore moins important.
- `<h5> </h5>` : titre pas important.
- `<h6> </h6>` : titre vraiment, mais alors là vraiment pas important du tout.



Attention : ne confondez pas avec la balise `<title>` ! La balise `<title>` affiche le titre de la page dans la barre de titre du navigateur comme nous l'avons vu. Les titres `<h1>` et compagnie, eux, servent à créer des titres qui seront affichés *dans* la page web.

Ne vous laissez pas impressionner par toutes ces balises. En fait, six niveaux de titres, c'est beaucoup. Dans la pratique, personnellement, je n'utilise que les balises `<h1>`, `<h2>` et `<h3>`, et très rarement les autres (je n'ai pas souvent besoin de six niveaux de titres différents). Votre navigateur affiche le titre très important en très gros, le titre un peu moins important en un peu moins gros, etc.



Ne choisissez pas votre balise de titre en fonction de la taille qu'elle applique au texte ! Il faut impérativement bien structurer sa page en commençant par un titre de niveau 1 (`<h1>`), puis un titre de niveau 2 (`<h2>`), etc. Il ne devrait pas y avoir de sous-titre sans titre principal ! Si vous voulez modifier la taille du texte, sachez que nous apprendrons à faire cela en CSS un peu plus tard.

Essayez de créer une page web avec des titres pour voir ce que cela donne :

```html
<html>
    <head>
        <meta charset="utf-8" />
        <title>Niveaux de titres</title>
    </head>

    <body>
        <h1>Titre super important</h1>
        <h2>Titre important</h2>
```

```
        <h3>Titre un peu moins important (sous-titre)</h3>

        <h4>Titre pas trop important</h4>
        <h5>Titre pas important</h5>
        <h6>Titre vraiment pas important du tout</h6>
    </body>
</html>
```

Allez, je vous donne un exemple d'utilisation des titres dans une page web (vous allez voir que je ne me sers pas de toutes les balises) :

```
<html>
<head>
    <meta charset="utf-8" />
    <title>Présentation du Site du Zéro</title>
</head>

<body>
    <h1>Bienvenue sur le Site du Zéro !</h1>

    <p>Bonjour et bienvenue sur mon site : le Site du Zéro.<br />
        Le Site du Zéro, qu'est-ce que c'est ?</p>

    <h2>Des cours pour débutants</h2>

    <p>Le Site du Zéro vous propose des cours (tutoriels) destinés aux débutants : aucune connaissance n'est requise pour lire ces cours !</p>

    <p>Vous pourrez ainsi apprendre, sans rien y connaître auparavant, à créer un site web, à programmer, à construire des mondes en 3D !</p>

    <h2>Une communauté active</h2>

    <p>Vous avez un problème, un élément du cours que vous ne comprenez pas ? Vous avez besoin d'aide pour créer votre site ?<br />
     Rendez-vous sur les forums ! Vous y découvrirez que vous n'êtes pas le seul dans ce cas et vous trouverez très certainement quelqu'un qui vous aidera aimablement à résoudre votre problème.</p>
</html>
```

▷ Copier ce code
Code web : 937479

Voilà une page web qui prend forme !



Oui, mais moi je veux centrer mon titre, l'écrire en rouge et le souligner !

Nous ferons tout cela lorsque nous apprendrons le CSS (dès la deuxième partie du cours). Il faut savoir que `<h1>` ne signifie pas « Times New Roman, taille 16 pt », mais *« Titre important »*.

Grâce au langage CSS, vous pourrez dire « Je veux que mes titres importants soient centrés, rouges et soulignés ». Pour le moment, en HTML, nous ne faisons que structurer notre page. *Nous rédigeons le contenu avant de nous amuser à le mettre en forme.*

La mise en valeur

Au sein de vos paragraphes, certains mots sont parfois plus importants que d'autres et vous aimeriez les faire ressortir. HTML vous propose différents moyens de mettre en valeur le texte de votre page.

Mettre un peu en valeur

Pour mettre *un peu* en valeur votre texte, vous devez utiliser la balise `<em> </em>`. Son utilisation est très simple : encadrez les mots à mettre en valeur avec ces balises et c'est bon ! Je reprends un peu l'exemple de tout à l'heure et j'y mets quelques mots en évidence :

```
<html>
<head>
    <meta charset="utf-8" />
    <title>Emphase</title>
</head>

<body>
    <p>Bonjour et bienvenue sur mon site !<br />
    Ceci est mon premier test alors <em>soyez indulgents</em> s'il vous plaît, j'apprends petit à petit comment cela marche.</p>
</body>
</html>
```

Comme vous pouvez le voir, utiliser la balise `<em>` a pour conséquence de mettre le texte en *italique*. En fait, c'est le navigateur qui choisit comment afficher les mots. On lui dit que les mots sont assez importants et, pour faire ressortir cette information, il change l'apparence du texte en utilisant l'italique.

Mettre bien en valeur

Pour mettre un texte bien en valeur, on utilise la balise `<strong>` qui signifie « fort », ou « important » si vous préférez. Elle s'utilise exactement de la même manière que `<em>` :

```
<html>
<head>
    <meta charset="utf-8" />
    <title>Forte emphase</title>
</head>

<body>
    <p>Bonjour et bienvenue sur mon site !</br />
    Ceci est mon premier test alors <strong>soyez indulgents</strong> s'il vous plaît, j'apprends petit à petit comment cela marche.</p>
</body>
</html>
```

Vous voyez sûrement le texte s'afficher en gras. Là encore, le gras n'est qu'une *conséquence*. Le navigateur a choisi d'afficher en gras les mots importants pour les faire ressortir davantage.

La balise `<strong>` ne signifie pas « mettre en gras » mais « important ». On pourra décider plus tard, en CSS, d'afficher les mots « importants » d'une autre façon que le gras si on le souhaite.

Marquer le texte

La balise `<mark>` permet de faire ressortir visuellement une portion de texte. L'extrait n'est pas forcément considéré comme important mais on veut qu'il se distingue bien du reste du texte. Cela peut être utile pour faire ressortir un texte pertinent après une recherche sur votre site par exemple.

```
<html>
<head>
    <meta charset="utf-8" />
    <title>Marquage du texte</title>
</head>

<body>
    <p>Bonjour et bienvenue sur mon site !<br />
    Ceci est mon premier test alors <mark>soyez indulgents</mark> s'il vous plaît, j'apprends petit à petit comment cela marche.</p>
</body>
</html>
```

Par défaut, `<mark>` a pour effet de surligner le texte. On pourra changer l'affichage en CSS (décider de surligner dans une autre couleur, d'encadrer le texte, etc.). C'est le même principe que ce que je vous disais pour les balises précédentes : elles indiquent le *sens* des mots et non pas comment ceux-ci doivent s'afficher.

N'oubliez pas : HTML pour le fond, CSS pour la forme

Je vais peut-être vous sembler un peu lourd mais il est très important qu'on se comprenne bien car les débutants font souvent la même grosse erreur à ce stade. Ils ont vu les balises `<em>`, `<strong>`, `<mark>`... et ils se disent : « Chouette, j'ai découvert comment mettre en italique, en gras et comment surligner du texte en HTML ! ».

Et pourtant... ce n'est pas à cela que servent ces balises ! Je sais, je sais, vous allez me dire « Oui mais quand j'utilise `<strong>` le texte apparaît en gras, donc c'est pour mettre en gras. » et pourtant, c'est une erreur de croire que cette balise sert à cela.

Le rôle des balises est d'indiquer le *sens* du texte. Ainsi, `<strong>` indique à l'ordinateur « Ce texte est important ». C'est tout. Et pour *montrer* que le texte est important, l'ordinateur décide de le mettre en gras (mais il pourrait aussi bien l'écrire en rouge !). La plupart des navigateurs affichent les textes importants en gras, mais rien ne les y oblige.



Je ne comprends pas. À quoi cela sert-il que l'ordinateur sache qu'un texte est important ? Il n'est pas assez intelligent pour comprendre !

Détrompez-vous ! De nombreux programmes analysent le code source des pages web, à commencer par les robots de moteurs de recherche. Ces robots parcourent le Web en lisant le code HTML de tous les sites. C'est le cas des robots de Google et de Bing, par exemple. Les mots-clés « importants » ont tendance à avoir plus de valeur à leurs yeux, donc si quelqu'un fait une recherche sur ces mots, il a plus de chances de tomber sur votre site. Bien entendu, c'est une explication grossière et il ne faut pas croire qu'utiliser la balise `<strong>` à tout-va améliorera votre référencement. Il faut simplement faire confiance aux ordinateurs : ils comprennent ce qu'un texte « important » veut dire et peuvent se servir de cette information.



Mais alors, comment fait-on pour mettre spécifiquement en gras, pour écrire en rouge, et tout et tout ?

Tout cela se fait en CSS. Souvenez-vous :

- le HTML définit le fond (contenu, logique des éléments) ;
- le CSS définit la forme (apparence).

Nous verrons le CSS plus loin, pour l'instant nous nous concentrons sur le HTML et ses balises, qui ont chacune un sens particulier.

Les listes

Les listes nous permettent souvent de mieux structurer notre texte et d'ordonner nos informations. Nous allons découvrir ici deux types de listes :

- les listes non ordonnées ou listes à puces ;
- les listes ordonnées ou listes numérotées ou encore énumérations.

Liste non ordonnée

Une liste non ordonnée ressemble à ceci :

- Fraises
- Framboises
- Cerises

C'est un système qui nous permet de créer une liste d'éléments sans notion d'ordre (il n'y a pas de « premier » ni de « dernier »). Créer une liste non ordonnée est très simple. Il suffit d'utiliser la balise `<ul>` que l'on referme un peu plus loin avec `</ul>`. Commencez donc à taper ceci :

```
<ul></ul>
```

Et maintenant, voilà ce qu'on va faire : on va écrire chacun des éléments de la liste entre deux balises `<li></li>`. Chacune de ces balises doit se trouver entre `<ul>` et `</ul>`. Vous allez comprendre de suite avec cet exemple :

```
<ul>
    <li>Fraises</li>
    <li>Framboises</li>
    <li>Cerises</li>
</ul>
```

Le résultat se trouve à la figure 3.2.

Notez que la liste doit être placée à l'intérieur de `<body></body>`. À partir de maintenant, je ne mets pas tout le code de la page pour rester lisible.

Retenez donc ces deux balises :

- `<ul></ul>` délimite toute la liste ;
- `<li></li>` délimite un élément de la liste (une puce).

Vous pouvez mettre autant d'éléments que vous voulez dans la liste à puces, vous n'êtes pas limités à trois éléments.

Et voilà, vous savez créer une liste à puces ! Pas si dur une fois qu'on a compris comment imbriquer les balises.

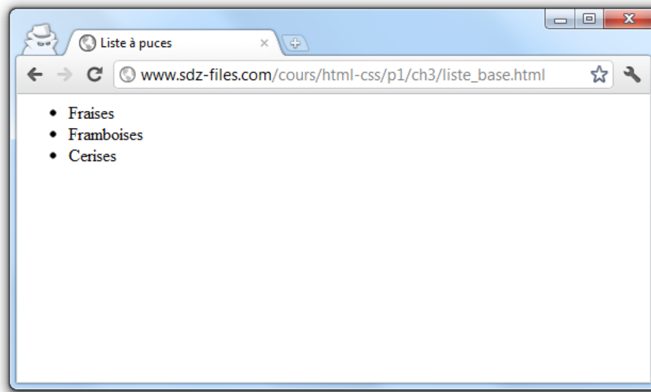


FIGURE 3.2 – Une liste non ordonnée



Pour ceux qui ont besoin de faire des listes complexes, sachez que vous pouvez *imbriquer* des listes à puces (créer une liste à puces **dans** une liste à puces). Si vous voulez faire ça, ouvrez une seconde balise `<ul>` **à l'intérieur** d'un élément `<li></li>`. Si vous fermez les balises dans le bon ordre, vous n'aurez pas de problème. Attention néanmoins, cette technique est un peu compliquée à maîtriser.

Liste ordonnée

Une liste ordonnée fonctionne de la même façon, seule une balise change : il faut remplacer `<ul></ul>` par `<ol></ol>`. À l'intérieur de la liste, on ne change rien : on utilise toujours des balises `<li></li>` pour délimiter les éléments.



L'ordre dans lequel vous placez les éléments de la liste est important. Le premier `<li></li>` sera l'élément n°1, le second sera le n°2 etc...

Comme c'est particulièrement intuitif, je vous laisse admirer la simplicité de cet exemple (résultat à la figure 3.3) :

```
<h1>Ma journée</h1>

<ol>
    <li>Je me lève.</li>
    <li>Je mange et je bois.</li>
    <li>Je retourne me coucher.</li>
</ol>
```

Par rapport à l'exemple précédent, tout ce qu'on a eu à changer est donc la balise `<ol>`.



FIGURE 3.3 – Une liste ordonnée



Pour information, il existe un troisième type de liste, beaucoup plus rare : la liste de définitions. Elle fait intervenir les balises `<dl>` (pour délimiter la liste), `<dt>` (pour délimiter un terme) et `<dd>` (pour délimiter la définition de ce terme).

En résumé

- Le HTML comporte de nombreuses balises qui nous permettent d'organiser le texte de notre page. Ces balises donnent des indications comme « Ceci est un paragraphe », « Ceci est un titre », etc.
- Les paragraphes sont définis par la balise `<p> </p>` et les sauts de ligne par la balise `<br />`.
- Il existe six niveaux de titre, de `<h1> </h1>` à `<h6> </h6>`, à utiliser selon l'importance du titre.
- On peut mettre en valeur certains mots avec les balises `<strong>`, `<em>` et `<mark>`.
- Pour créer des listes, on doit utiliser la balise `<ul>` (liste à puces, non ordonnée) ou `<ol>` (liste ordonnée). À l'intérieur, on insère les éléments avec une balise `<li>` pour chaque item.

Chapitre 4

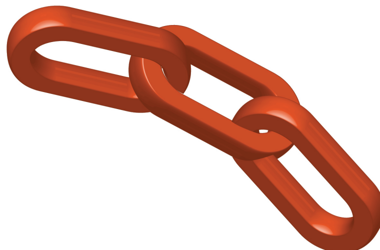
Créer des liens

Difficulté :

Au chapitre précédent, vous avez appris à créer une page HTML toute simple. D'accord, elle n'était pas franchement magnifique, mais c'était une vraie page HTML quand même.

Comme vous le savez, un site web est composé de plusieurs pages. Comment faire pour aller d'une page vers une autre ? À l'aide de liens pardi ! Dans ce chapitre, nous allons justement apprendre à créer des liens entre nos pages.

Je suppose que chacun d'entre vous sait ce qu'est un lien : il s'agit d'un texte sur lequel on peut cliquer pour se rendre sur une autre page. On peut faire un lien d'une page `a.html` vers une page `b.html`, mais on peut aussi faire un lien vers un autre site (par exemple, `http://www.siteduzero.com`). Dans les deux cas, nous allons voir que le fonctionnement est le même.



Un lien vers un autre site

Il est facile de reconnaître les liens sur une page : ils sont écrits d'une façon différente (par défaut, en bleu et soulignés) et un curseur en forme de main apparaît lorsqu'on pointe dessus.

Je vous propose d'essayer de coder le lien qui amène vers le Site du Zéro, comme à la figure 4.1.

Bonjour. Souhaitez-vous visiter le Site du Zéro ?
C'est un bon site ! ;-)

FIGURE 4.1 – Lien vers le Site du Zéro

Pour faire un lien, la balise que nous allons utiliser est très simple à retenir : `<a>`. Il faut cependant lui ajouter un attribut, `href`, pour indiquer vers quelle page le lien doit conduire.

Par exemple, le code ci-dessous est un lien qui amène vers le Site du Zéro, situé à l'adresse `http://www.siteduzero.com` :

```
<a href="http://www.siteduzero.com">Site du Zéro</a>
```

Nous allons placer ce lien au sein d'un paragraphe. Voici donc comment reproduire l'exemple de la figure 4.1 :

```
<p>Bonjour. Souhaitez-vous visiter le <a href="http://www.
    siteduzero.com">Site du Zéro</a> ?<br />
C'est un bon site ! ;-)</p>
```



Par défaut, le lien s'affiche en bleu souligné. Si vous avez déjà ouvert la page, le lien s'affiche en violet. Nous verrons comment changer cette apparence lorsque nous étudierons le CSS.

Si vous voulez faire un lien vers un autre site, il suffit donc de copier son adresse (on parle d'URL) en `http://`. Notez que certains liens commencent parfois par `https://` (sites sécurisés) ou d'autres préfixes (`ftp://`,...).



Si vous faites un lien vers un site qui comporte une adresse un peu bizarre avec des &, comme : `http://www.site.com/?data=15&name=mateo21`, vous devrez remplacer tous les « & » par « & » dans votre lien comme ceci : `http://www.site.com/?data=15&name=mateo21`. Vous ne verrez pas la différence, mais cela est nécessaire pour avoir une page web correctement construite en HTML5.

Les liens que nous venons de voir sont appelés **liens absolus** car on indique l'adresse complète. Nous allons maintenant voir que l'on peut écrire les liens d'une façon un peu différente, ce qui va nous être utile pour faire des liens entre les pages de notre site.

Un lien vers une autre page de son site

Nous venons d'apprendre à créer des liens vers des sites existants. Mais je suis sûr que vous aimeriez faire des liens entre les différentes pages de votre site, non ?



Oui, justement, comment je fais pour faire un lien vers une autre page de mon site ? Je ne connais pas son adresse en http://..., je commence à peine à créer mon site là ! Je n'ai pas d'adresse.

En effet, pour le moment, vous êtes en train de créer votre site sur votre ordinateur. Vous êtes le seul à pouvoir le voir et il n'a pas encore « d'adresse web » qui commence en http:// comme la plupart des sites. Heureusement, cela ne va pas nous empêcher de travailler.

Deux pages situées dans un même dossier

Pour commencer, nous allons créer deux fichiers correspondant à deux pages HTML différentes. Comme je suis très inspiré, je vous propose de les appeler page1.html et page2.html. Nous aurons donc ces deux fichiers sur notre disque *dans le même dossier* (figure 4.2).

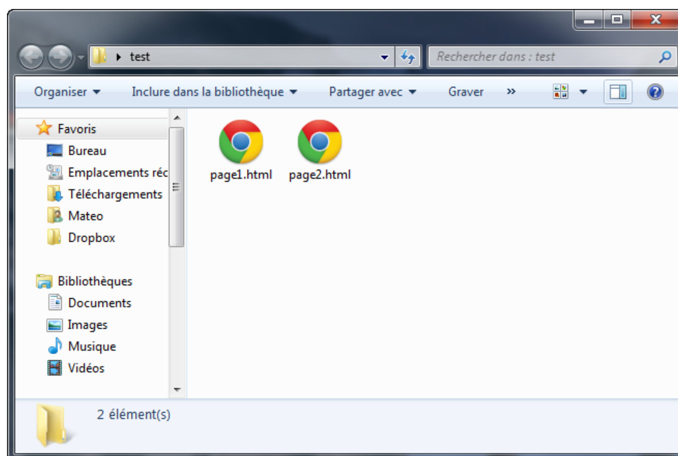


FIGURE 4.2 – Plusieurs fichiers HTML dans un même dossier

Comment faire un lien de la page 1 vers la page 2, sans avoir d'adresse en http:// ? En fait, c'est facile : si les deux fichiers sont situés dans le même dossier, il suffit d'écrire comme cible du lien le nom du fichier vers lequel on veut amener. Par exemple : <a href="page2.html">. On dit que c'est un **lien relatif**.

Voici le code que nous allons utiliser dans nos fichiers page1.html et page2.html.

`page1.html`

```
<p>Bonjour. Souhaitez-vous consulter <a href="page2.html">la
    page 2</a> ?</p>
```

`page2.html`

La page 2 (page d'arrivée) affichera simplement un message pour indiquer que l'on est bien arrivé sur la page 2 :

```
<h1>Bienvenue sur la page 2 !</h1>
```

Deux pages situées dans des dossiers différents

Les choses se corsent un petit peu si les pages sont situées dans des dossiers différents. Idéalement, elles ne devraient pas être trop loin l'une de l'autre (dans un sous-dossier par exemple).

Imaginons que `page2.html` se trouve dans un sous-dossier appelé `contenu`, comme à la figure 4.3.

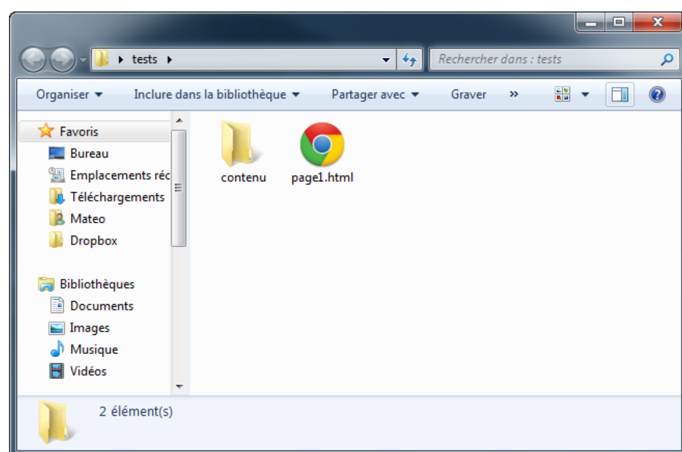


FIGURE 4.3 – Le fichier `page2.html` se trouve à l'intérieur du dossier `contenu`

Dans ce cas de figure, le lien doit être rédigé comme ceci :

```
<a href="contenu/page2.html">
```

S'il y avait plusieurs sous-dossiers, on écrirait ceci :

```
<a href="contenu/autredossier/page2.html">
```



Et si le fichier ne se trouve pas dans un sous-dossier mais dans un dossier « parent », on fait comment ?

Si votre fichier cible est placé dans un dossier qui se trouve « plus haut » dans l'arborescence, il faut écrire deux points comme ceci :

```
<a href="../page2.html">
```

Résumé en images

Les liens relatifs ne sont pas bien compliqués à utiliser une fois qu'on a compris le principe. Il suffit de regarder dans quel « niveau de dossier » se trouve votre fichier cible pour savoir comment écrire votre lien. La figure 4.4 fait la synthèse des différents liens relatifs possibles.

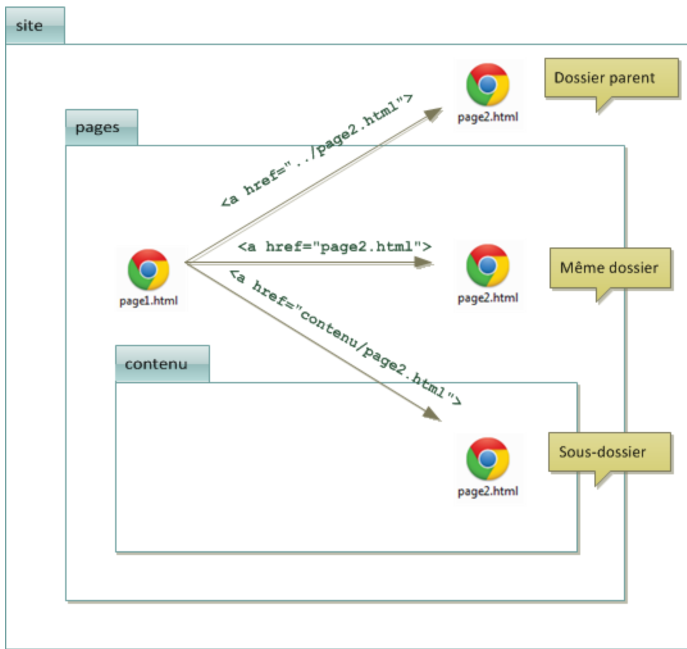


FIGURE 4.4 – Les différents liens relatifs

Un lien vers une ancre

Une **ancre** est une sorte de point de repère que vous pouvez mettre dans vos pages HTML lorsqu'elles sont très longues. En effet, il peut alors être utile de faire un lien amenant plus bas dans la même page pour que le visiteur puisse sauter directement à la partie qui l'intéresse.

Pour créer une ancre, il suffit de rajouter l'attribut `id` à une balise qui va alors servir de repère. Ce peut être n'importe quelle balise, un titre par exemple. Utilisez l'attribut

`id` pour donner un nom à l'ancre. Cela nous servira ensuite pour faire un lien vers cette ancre. Par exemple :

```
<h2 id="mon_ancre">Titre</h2>
```

Ensuite, il suffit de créer un lien comme d'habitude, mais cette fois l'attribut `href` contiendra un dièse (#) suivi du nom de l'ancre. Exemple :

```
<a href="#mon_ancre">Aller vers l'ancre</a>
```

Normalement, si vous cliquez sur le lien, cela vous amènera plus bas dans la même page (à condition que la page comporte suffisamment de texte pour que les barres de défilement se déplacent automatiquement). Voici un exemple de page comportant beaucoup de texte et utilisant les ancres (j'ai mis n'importe quoi dans le texte pour remplir) :

```
<h1>Ma grande page</h1>

<p>
    Aller directement à la partie traitant de :<br />
    <a href="#cuisine">La cuisine</a><br />
    <a href="#rollers">Les rollers</a><br />
    <a href="#arc">Le tir à l'arc</a><br />
</p>
<h2 id="cuisine">La cuisine</h2>

<p>... (beaucoup de texte) ...</p>

<h2 id="rollers">Les rollers</h2>

<p>... (beaucoup de texte) ...</p>

<h2 id="arc">Le tir à l'arc</h2>

<p>... (beaucoup de texte) ...</p>
```

S'il ne se passe rien quand vous cliquez sur les liens, c'est qu'il n'y a pas assez de texte. Dans ce cas, vous pouvez soit rajouter du blabla dans la page pour qu'il y ait (encore) plus de texte, soit réduire la taille de la fenêtre de votre navigateur pour faire apparaître les barres de défilement sur le côté.



L'attribut `id` sert à donner un nom « unique » à une balise, pour s'en servir de repère. Et, croyez-moi, vous n'avez pas fini d'entendre parler de cet attribut. Ici, on s'en sert pour faire un lien vers une ancre mais, en CSS, il nous sera très utile pour « repérer » une balise précise, vous verrez. Évitez cependant de créer des id avec des espaces ou des caractères spéciaux, utilisez simplement, dans la mesure du possible, des lettres et chiffres pour que la valeur soit reconnue par tous les navigateurs.

Lien vers une ancre située dans une autre page

Alors là je vous préviens, on va faire le Mégamix !

L'idée, c'est de faire un lien qui ouvre une autre page ET qui amène directement à une ancre située plus bas sur cette page. En pratique c'est assez simple à faire : il suffit de taper le nom de la page, suivi d'un dièse (#), suivi du nom de l'ancre.

Par exemple : `<a href="ancres.html#rollers">`

... vous amènera sur la page `ancres.html`, directement au niveau de l'ancre appelée `rollers`.

Voici une page qui contient trois liens, chacun amenant vers une des ancres de la page de l'exemple précédent :

```
<h1>Le Mégamix</h1>
<p>
    Rendez-vous quelque part sur une autre page :<br />
    <a href="ancres.html#cuisine">La cuisine</a><br />
    <a href="ancres.html#rollers">Les rollers</a><br />
    <a href="ancres.html#arc">Le tir à l'arc</a><br />
</p>
```

Cas pratiques d'utilisation des liens

Je vais essayer de vous montrer ici quelques cas pratiques d'utilisation des liens. Par exemple, saviez-vous qu'il est très facile de faire des liens qui lancent un téléchargement ? Qui créent un nouvel e-mail ? Qui ouvrent une nouvelle fenêtre ?

Non ? Eh bien nous allons voir tout cela ici.

Un lien qui affiche une infobulle au survol

Vous pouvez utiliser l'attribut `title` qui affiche une bulle d'aide lorsqu'on pointe sur le lien. Cet attribut est facultatif.

Vous aurez un résultat ressemblant à la figure 4.5.



FIGURE 4.5 – Une infobulle

La bulle d'aide peut être utile pour informer le visiteur avant même qu'il n'ait cliqué sur le lien. Voici comment reproduire ce résultat :

```
<p>Bonjour. Souhaitez-vous visiter le <a href="http://www.
    siteduzero.com" title="Réservé aux débutants">Site du Zéro</
    a> ?</p>
```

Un lien qui ouvre une nouvelle fenêtre

Il est possible de « forcer » l'ouverture d'un lien dans une nouvelle fenêtre. Pour cela, on rajoutera `target="_blank"` à la balise `<a>` :

```
<p>Bonjour. Souhaitez-vous visiter le <a href="http://www.siteduzero.com" target="_blank">Site du Zéro</a> ?<br />
Le site s'affichera dans une autre fenêtre.</p>
```



Selon la configuration du navigateur, la page s'affichera dans une nouvelle fenêtre ou un nouvel onglet. Vous ne pouvez pas choisir entre l'ouverture d'une nouvelle fenêtre ou d'un nouvel onglet.



Notez cependant qu'il est déconseillé d'abuser de cette technique car elle perturbe la navigation. Le visiteur lui-même peut décider s'il veut ouvrir le lien dans une nouvelle fenêtre. Il fera `Maj` + Clic sur le lien pour ouvrir dans une nouvelle fenêtre ou `Ctrl` + Clic pour ouvrir dans un nouvel onglet.

Un lien pour envoyer un e-mail

Si vous voulez que vos visiteurs puissent vous envoyer un e-mail, vous pouvez utiliser des liens de type `mailto`. Rien ne change au niveau de la balise, vous devez simplement modifier la valeur de l'attribut `href` comme ceci :

```
<p><a href="mailto:votrenom@bidule.com">Envoyez-moi un e-mail !</a></p>
```

Il suffit donc de faire commencer le lien par `mailto:` et d'écrire l'adresse e-mail où on peut vous contacter. Si vous cliquez sur le lien, un nouveau message vide s'ouvre, prêt à être envoyé à votre adresse e-mail.

Un lien pour télécharger un fichier

Beaucoup d'entre vous se demandent comment cela se passe pour le téléchargement d'un fichier... En fait, il faut procéder exactement comme si vous faisiez un lien vers une page web, mais en indiquant cette fois le nom du fichier à télécharger.

Par exemple, supposez que vous vouliez faire télécharger `monfichier.zip`. Placez simplement ce fichier dans le même dossier que votre page web (ou dans un sous-dossier) et faites un lien vers ce fichier :

```
<p><a href="monfichier.zip">Télécharger le fichier</a></p>
```

C'est tout ! Le navigateur, voyant qu'il ne s'agit pas d'une page web à afficher, va lancer la procédure de téléchargement lorsqu'on cliquera sur le lien.

En résumé

- Les liens permettent de changer de page et sont, par défaut, écrits en bleu et soulignés.
- Pour insérer un lien, on utilise la balise `<a>` avec l'attribut `href` pour indiquer l'adresse de la page cible. Exemple : `<a href="http://www.siteduzero.com">`.
- On peut faire un lien vers une autre page de son site simplement en écrivant le nom du fichier : `<a href="page2.html">`.
- Les liens permettent aussi d'amener vers d'autres endroits sur la même page. Il faut créer une ancre avec l'attribut `id` pour « marquer » un endroit dans la page, puis faire un lien vers l'ancre comme ceci : `<a href="#ancre">`.

Chapitre 5

Les images

Difficulté :

Insérer une image dans une page web ? Vous allez voir, c'est d'une facilité déconcertante... Enfin presque. Il existe différents **formats** d'image que l'on peut utiliser sur des sites web, et on ne doit pas les choisir au hasard. En effet, les images sont parfois volumineuses à télécharger, ce qui ralentit le temps de chargement de la page (beaucoup plus que le texte !).

Pour faire en sorte que vos pages restent lisibles et rapides à télécharger, suivez donc activement mes conseils !



Les différents formats d'images

Savez-vous ce qu'est un format d'image ?

Quand vous avez une image « entre les mains », vous avez la possibilité de l'enregistrer dans plusieurs « formats » différents. Le poids (en Ko, voire en Mo) de l'image sera plus ou moins élevé selon le format choisi et la qualité de l'image va changer.

Par exemple, le logiciel de dessin Paint (même si c'est loin d'être le meilleur) vous propose de choisir entre plusieurs formats lorsque vous enregistrez une image (figure 5.1).

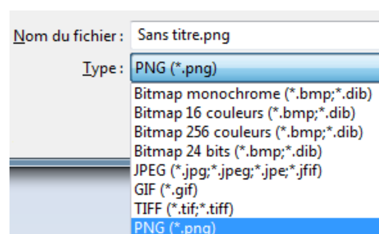


Figure 5.1 – Différents formats d'image proposés par Paint

Certains formats sont plus adaptés que d'autres selon l'image (photo, dessin, image animée...). Notre but ici est de faire le tour des différents formats utilisés sur le Web pour que vous les connaissiez et sachiez choisir celui qui convient le mieux à votre image. Rassurez-vous, il n'y a pas beaucoup de formats différents, cela ne sera donc pas bien long.

Toutes les images diffusées sur Internet ont un point commun : elles sont **compressées**. Cela veut dire que l'ordinateur fait des calculs pour qu'elles soient moins lourdes et donc plus rapides à charger.

Le JPEG

Les images au format JPEG (*Joint Photographic Expert Group*) sont très répandues sur le Web. Ce format est conçu pour réduire le poids des photos (c'est-à-dire la taille du fichier associé), qui peuvent comporter plus de 16 millions de couleurs différentes. La figure 5.2 est une photo enregistrée au format JPEG.

Les images JPEG sont enregistrées avec l'extension `.jpg` ou `.jpeg`.

Notez que le JPEG détériore un peu la qualité de l'image, d'une façon généralement imperceptible. C'est ce qui le rend si efficace pour réduire le poids des photos. Quand il s'agit d'une photo, on ne peut généralement pas détecter la perte de qualité. Par contre, si ce n'est pas une photo, vous risquez de voir l'image un peu « baver ». Dans ce cas, il vaut mieux utiliser le format PNG.



FIGURE 5.2 – Une photo de montagne en JPEG

Le PNG

Le format PNG (*Portable Network Graphics*) est le plus récent de tous. Ce format est adapté à la plupart des graphiques (je serais tenté de dire « à tout ce qui n'est pas une photo »). Le PNG a deux gros avantages : il peut être rendu transparent et il n'altère pas la qualité de l'image.

Le PNG a été inventé pour concurrencer un autre format, le GIF, à l'époque où il fallait payer des royalties pour pouvoir utiliser des GIF. Depuis, le PNG a bien évolué et c'est devenu le format le plus puissant pour enregistrer la plupart des images.

Le PNG existe en deux versions, en fonction du nombre de couleurs que doit comporter l'image :

- PNG 8 bits : 256 couleurs ;
- PNG 24 bits : 16 millions de couleurs (autant qu'une image JPEG).

La figure 5.3 est une image PNG en 24 bits, représentant la célèbre mascotte Zozor du Site du Zéro.



FIGURE 5.3 – Zozor en PNG



Une vieille version d'Internet Explorer (IE6) ne peut pas afficher correctement les images PNG 24 bits transparentes. Ce navigateur tend à être de moins en moins utilisé, mais gardez quand même cette information en tête.



Au fait, si le PNG 24 bits peut afficher autant de couleurs qu'une image JPEG, et qu'en plus il peut être rendu transparent sans modifier la qualité de l'image... quel est l'intérêt du JPEG ?

La compression du JPEG est plus puissante sur les photos. Une photo enregistrée en JPEG se chargera toujours beaucoup plus vite que si elle était enregistrée en PNG. Je vous conseille donc toujours de réserver le format JPEG aux photos.

Le GIF

C'est un format assez vieux, qui a été néanmoins très utilisé (et qui reste très utilisé par habitude). Aujourd'hui, le PNG est globalement bien meilleur que le GIF : les images sont généralement plus légères et la transparence est de meilleure qualité. Je vous recommande donc d'utiliser le PNG autant que possible.

Le format GIF est limité à 256 couleurs (alors que le PNG peut aller jusqu'à plusieurs millions de couleurs).

Néanmoins, le GIF conserve un certain avantage que le PNG n'a pas : il peut être animé.

▷ Un GIF animé
Code web : 280031

Il existe un format adapté à chaque image

Si on résume, voici quel format adopter en fonction de l'image que vous avez :

- **Une photo** : utilisez un JPEG.
- **N'importe quel graphique avec peu de couleurs** (moins de 256) : utilisez un PNG 8 bits ou éventuellement un GIF.
- **N'importe quel graphique avec beaucoup de couleurs** : utilisez un PNG 24 bits.
- **Une image animée** : utilisez un GIF animé.

Les erreurs à éviter

Bannissez les autres formats

Les autres formats non cités ici, comme le format BITMAP (`*.bmp`) sont à bannir car bien souvent ils ne sont pas compressés, donc trop gros. Ils ne sont pas du tout adaptés au Web. On peut en mettre sur son site mais le chargement sera vraiment *extrêmement*

long !

Choisissez bien le nom de votre image

Si vous voulez éviter des problèmes, prenez l'habitude d'enregistrer vos fichiers avec des noms en minuscules, sans espace ni accent, par exemple : `mon_image.png`.

Vous pouvez remplacer les espaces par le caractère *underscore* (« _ ») comme je l'ai fait ici.

Insérer une image

Revenons maintenant au code HTML pour découvrir comment placer des images dans nos pages web !

Insertion d'une image

Quelle est la fameuse balise qui va nous permettre d'insérer une image ? Il s'agit de... `<img />` !

C'est une balise de type **orpheline** (comme `<br />`). Cela veut dire qu'on n'a pas besoin de l'écrire en deux exemplaires comme la plupart des autres balises que nous avons vues jusqu'ici. En effet, nous n'avons pas besoin de délimiter une portion de texte, nous voulons juste insérer une image à un endroit précis.

La balise doit être accompagnée de deux attributs obligatoires :

- `src` : il permet d'indiquer où se trouve l'image que l'on veut insérer. Vous pouvez soit mettre un chemin absolu (ex. : `http://www.site.com/fleur.png`), soit mettre le chemin en relatif (ce qu'on fait le plus souvent). Ainsi, si votre image est dans un sous-dossier `images`, vous devrez taper : `src="images/fleur.png"`
- `alt` : cela signifie « texte alternatif ». On doit *toujours* indiquer un texte alternatif à l'image, c'est-à-dire un court texte qui décrit ce que contient l'image. Ce texte sera affiché à la place de l'image si celle-ci ne peut pas être téléchargée (cela arrive), ou dans les navigateurs de personnes handicapées (non-voyants) qui ne peuvent malheureusement pas « voir » l'image. Cela aide aussi les robots des moteurs de recherche pour les recherches d'images. Pour la fleur, on mettrait par exemple : `alt="Une fleur"`.

Les images doivent se trouver obligatoirement à l'intérieur d'un paragraphe (`<p></p>`). Voici un exemple d'insertion d'image :

```
<p>
    Voici une photo que j'ai prise lors de mes dernières
        vacances à la montagne :<br />
    <img src="images/montagne.jpg" alt="Photo de montagne" />
</p>
```

Bref, l'insertion d'image est quelque chose de très facile pour peu qu'on sache indiquer où se trouve l'image, comme on avait appris à le faire avec les liens. La plus grosse « difficulté » (si on peut appeler cela une difficulté) consiste à choisir le bon format d'image. Ici, c'est une photo donc c'est évidemment le format JPEG qu'on utilise.



Je le répète : évitez à tout prix les accents, majuscules et espaces dans vos noms de fichiers et de dossiers. Voici un chemin qui va poser problème : "Images du site/Image toute bête.jpg". Il faudrait supprimer les espaces (ou les remplacer par le symbole « _ »), supprimer les accents et tout mettre en minuscules comme ceci : "images_du_site/image_toute_bete.jpg". Sachez donc que, si votre image ne s'affiche pas, c'est très certainement parce que le chemin est incorrect ! Simplifiez au maximum vos noms de fichiers et de dossiers, et tout ira bien.

Ajouter une infobulle

L'attribut permettant d'afficher une bulle d'aide est le même que pour les liens : il s'agit de `title`. Cet attribut est facultatif (contrairement à `alt`).

Voici ce que cela peut donner :

```
<p>
    Voici une photo que j'ai prise lors de mes dernières
        vacances à la montagne :<br />
    <img src="images/montagne.jpg" alt="Photo de montagne"
        title="C'est beau les Alpes quand même !" />
</p>
```

Survolez la photo avec la souris pour voir l'infobulle apparaître.

Miniature cliquable

Si votre image est très grosse, il est conseillé d'en afficher la miniature sur votre site. Ajoutez ensuite un lien sur cette miniature pour que vos visiteurs puissent afficher l'image en taille originale.

Il existe des millions de logiciels permettant de créer des miniatures d'images. J'utilise personnellement Easy Thumbnails. Je vais ainsi disposer de deux versions de ma photo, comme à la figure 5.4) : la miniature et l'image d'origine.

▷ Easy Thumbnails
Code web : 663893

Je les place toutes les deux dans un dossier appelé par exemple `img`. J'affiche la version `montagne_mini.jpg` sur ma page et je fais un lien vers `montagne.jpg` pour que l'image agrandie s'affiche lorsqu'on clique sur la miniature.

Voici le code HTML que je vais utiliser pour cela :

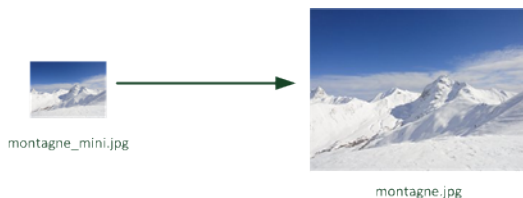


FIGURE 5.4 – La miniature et son image d'origine

```
<p>
    Vous souhaitez voir l'image dans sa taille d'origine ?
        Cliquez dessus !<br />
    <a href="img/montagne.jpg"><img src="img/montagne_mini.jpg"
         alt="Photo de montagne" title="Cliquez pour agrandir" /
        ></a>
</p>
```



Parfois, certains navigateurs choisissent d'afficher un cadre bleu (ou violet) pas très esthétique autour de votre image cliquable. Heureusement, nous pourrons retirer ce cadre dans peu de temps grâce au CSS.

Les figures

Au cours de la lecture de ce livre, vous avez déjà rencontré plusieurs fois des **figures**. Ce sont des éléments qui viennent enrichir le texte pour compléter les informations de la page.

Les figures peuvent être de différents types :

- images ;
- codes source ;
- citations ;
- etc.

Bref, tout ce qui vient *illustrer* le texte est une figure. Nous allons ici nous intéresser aux images mais, contrairement à ce qu'on pourrait croire, les figures ne sont pas *forcément* des images : un code source aussi illustre le texte.

Création d'une figure

Reprenons par exemple cette capture d'écran du premier chapitre, représentée à la figure 5.5.

En HTML5, on dispose de la balise `<figure>`. Voici comment on pourrait l'utiliser :

```
<figure>
```

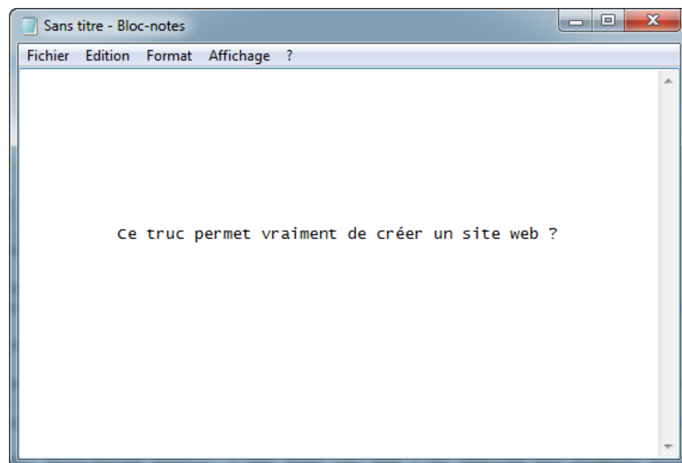


Figure 5.5 – Le logiciel Bloc-Notes

```
    <img src="images/blocnotes.png" alt="Bloc-Notes" />
</figure>
```

Une figure est le plus souvent accompagnée d'une légende. Pour ajouter une légende, utilisez la balise `<figcaption>` à l'intérieur de la balise `<figure>`, comme ceci :

```
<figure>
    <img src="images/blocnotes.png" alt="Bloc-Notes" />
    <figcaption>Le logiciel Bloc-Notes</figcaption>
</figure>
```

Bien comprendre le rôle des figures

Un peu plus tôt dans ce chapitre, je vous ai dit que les images devaient être situées dans des paragraphes (placées à l'intérieur d'une balise `<p></p>`). Ce n'est pas tout à fait vrai.

Si vous faites de votre image une figure, l'image peut être située en-dehors d'un paragraphe.

```
<p>Connaissez-vous le logiciel Bloc-Notes ? On peut faire des
   sites web avec !</p>

<figure>
    <img src="images/blocnotes.png" alt="Bloc-Notes" />
    <figcaption>Le logiciel Bloc-Notes</figcaption>
</figure>
```



Je ne vois pas vraiment de changement. Quand dois-je placer mon image dans un paragraphe et quand dois-je la placer dans une figure ?

Bonne question ! Tout dépend de ce que votre image apporte au texte :

- Si elle n'apporte aucune information (c'est juste une illustration pour décorer) : placez l'image dans un paragraphe.
- Si elle apporte une information : placez l'image dans une figure.

La balise `<figure>` a un rôle avant tout **sémantique**. Cela veut dire qu'elle indique à l'ordinateur que l'image a du sens et qu'elle est importante pour la bonne compréhension du texte. Cela peut permettre à un programme de récupérer toutes les figures du texte et de les référencer dans une table des figures, par exemple.

Enfin, sachez qu'une figure peut très bien comporter plusieurs images. Voici un cas où cela se justifie :

```
<figure>
    <img src="images/internetexplorer.png" alt="Logo Internet Explorer" />
    <img src="images/firefox.png" alt="Logo Mozilla Firefox" />
    <img src="images/chrome.png" alt="Logo Google Chrome" />
    <figcaption>Logos des différents navigateurs</figcaption>
</figure>
```

En résumé

- Il existe plusieurs formats d'images adaptées au Web :
  - JPEG : pour les photos ;
  - PNG : pour toutes les autres illustrations ;
  - GIF : similaire au PNG, plus limité en nombre de couleurs mais qui peut être animé.
- On insère une image avec la balise `<img />`. Elle doit obligatoirement comporter au moins ces deux attributs : `src` (nom de l'image) et `alt` (courte description de l'image).
- Si une image illustre le texte (et n'est pas seulement décorative), il est conseillé de la placer au sein d'une balise `<figure>`. La balise `<figcaption>` permet d'écrire la légende de l'image.

Deuxième partie

Les joies de la mise en forme avec CSS

Chapitre 6

Mettre en place le CSS

Difficulté :

Après avoir passé toute une première partie du cours à ne travailler que sur le HTML, nous allons maintenant découvrir le CSS que j'avais volontairement mis à l'écart. Le CSS n'est pas plus compliqué que le HTML. Il vient le compléter pour vous aider à mettre en forme votre page web.

Dans ce premier chapitre sur le CSS, nous allons voir la théorie sur le CSS : qu'est-ce que c'est ? À quoi cela ressemble-t-il ? Où est-ce qu'on écrit du code CSS ? Ces aspects théoriques ne sont pas bien compliqués mais vous devez obligatoirement les connaître car c'est la base du CSS. C'est d'ailleurs la seule chose que je vous demanderai de retenir par cœur en CSS, vous pourrez retrouver le reste dans le mémo en annexe.

Allez, ne traînons pas, je vois que vous brûlez d'impatience !



La petite histoire du CSS

Je vous avais avertis dès le début de ce cours : nous allons apprendre deux langages. Nous avons déjà bien entamé notre découverte du HTML, même s'il reste encore de nombreuses choses à apprendre (nous y reviendrons dans quelques chapitres). En revanche, il est temps maintenant de nous intéresser au CSS.

CSS (*Cascading Style Sheets*), c'est cet autre langage qui vient compléter le HTML. Vous vous souvenez de son rôle ? *Gérer la mise en forme de votre site.*

Petit rappel : à quoi sert CSS ?

CSS ? C'est lui qui vous permet de choisir la couleur de votre texte. Lui qui vous permet de sélectionner la police utilisée sur votre site. Lui encore qui permet de définir la taille du texte, les bordures, le fond...

Et aussi, c'est lui qui permet de faire la mise en page de votre site. Vous pourrez dire : je veux que mon menu soit à gauche et occupe telle largeur, que l'en-tête de mon site soit calé en haut et qu'il soit toujours visible, etc.

Souvenez-vous de ce petit comparatif que nous avions vu dès le premier chapitre (figure 6.1).

HTML
(pas de CSS)



HTML + CSS



Figure 6.1 – La même page HTML, sans et avec CSS (`www.csszengarden.com`)

Grâce au HTML, nous avons pu rédiger le contenu de notre site mais il est brut de décoffrage. Le CSS vient compléter ce code pour mettre en forme tout cela et donner

au contenu l'apparence que l'on souhaite.

CSS : des débuts difficiles

Il faut savoir qu'aux débuts du Web, CSS n'existait pas. En fait, il n'y avait initialement que le langage HTML. Le HTML est né en 1991 et CSS en 1996. Alors, vous vous dites sûrement : comment faisait-on la mise en forme de 1991 à 1996 ? Eh bien, uniquement en HTML ! Il y avait en effet des balises HTML dédiées à la mise en forme. `<font color="#aab1c3">`, par exemple, permettait de définir la couleur du texte.

Cependant, les pages HTML commençaient à devenir assez complexes. Il y avait de plus en plus de balises et c'était un joyeux mélange entre le fond et la forme, qui rendait la mise à jour des pages web de plus en plus complexe. C'est pour cela que l'on a créé le langage CSS.

Cependant, le CSS n'a pas été adopté immédiatement par les webmasters, loin de là. Il fallait se défaire de certaines mauvaises habitudes et cela a pris du temps. Encore aujourd'hui, on peut trouver des sites web avec des balises HTML de mise en forme, anciennes et obsolètes, comme `<font>` !

CSS : la prise en charge des navigateurs

Tout comme le HTML, le CSS a évolué. Je vous avais indiqué qu'il y avait quatre versions importantes de CSS :

- CSS 1 ;
- CSS 2.0 ;
- CSS 2.1 ;
- CSS 3.



En fait, la version CSS 3 n'est pas encore totalement finalisée (ce n'est pas encore une version officielle). Cependant, elle est bien avancée et aujourd'hui déjà bien prise en charge par de nombreux navigateurs, ce qui fait qu'on peut déjà s'en servir. Il serait dommage de passer à côté car CSS 3 apporte de nombreuses fonctionnalités à CSS (leur nombre double par rapport à CSS 2.1 !). Nous nous baserons donc dans ce cours sur CSS 3, qui reprend et complète la plupart des fonctionnalités de CSS 2.1.

Ce sont les navigateurs web qui font le travail le plus complexe : ils doivent *lire* le code CSS et *comprendre* comment afficher la page.

Au début des années 2000, Internet Explorer était le navigateur le plus répandu mais sa gestion du CSS est longtemps restée assez médiocre (pour ne pas dire carrément mauvaise). C'était la grande époque de la version 6 (IE6), hélas encore utilisée aujourd'hui par une petite partie des internautes (heureusement, cette proportion tend à diminuer).

Depuis, de nombreux navigateurs sont arrivés et ont chahuté Internet Explorer : Mozilla Firefox bien sûr, mais aussi Google Chrome. Et je ne vous parle pas du succès des Mac et iPhone avec leur navigateur Safari. Cela a incité Microsoft à réagir et publier (après une longue période d'inactivité) IE 7, puis IE 8 et IE 9. On parle déjà de IE 10.



Bon, ton cours d'histoire, c'est bien joli mais en quoi cela me concerne-t-il aujourd'hui ?

Que faut-il retenir de tout cela ? Que les navigateurs ne connaissent pas toutes les propriétés CSS qui existent. Plus le navigateur est vieux, moins il connaît de fonctionnalités CSS.

Je vais vous présenter dans ce cours un certain nombre de fonctionnalités de CSS qui ne marchent pas forcément sur les navigateurs les plus vieux. Je ne peux pas l'éviter, c'est comme cela : *aucun navigateur ne connaît parfaitement toutes les fonctionnalités CSS* de toute façon ! Au pire, si le navigateur ne connaît pas une propriété CSS, il l'ignore et ne met pas en forme, mais cela ne fait pas planter votre page : celle-ci sera donc toujours lisible.

Je vous recommande fortement de mettre dans vos favoris les sites `www.caniuse.com` et `www.normansblog.de` qui proposent des tables de compatibilité des fonctionnalités de HTML et CSS sur différents navigateurs (et sur leurs différentes versions). Regardez en particulier les tables de compatibilité pour CSS de `www.caniuse.com` (figure 6.2).

CSS position:fixed - Recommendation

Method of keeping an element in a fixed location regardless of scroll position

Resources: Workaround for IE6 Workaround for Mobile Safari

Global user stats*:

Support:	88.51%
Partial support:	1.14%
Total:	89.65%

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser
3 versions back	6.0	4.0	11.0	3.2	10.6	3.2			
2 versions back	7.0	5.0	12.0	4.0	11.0	4.0-4.1		10.0	2.1
Previous version	8.0	6.0	13.0	5.0	11.1	4.2-4.3		11.0	2.2
Current	9.0	7.0	14.0	5.1	11.5	5.0	5.0-6.0	11.1	2.3 / 3.0
Near future		8.0	15.0		12.0				4.0
Farther future	10.0	9.0	16.0	6.0	12.1				

Note: Only works in Android 2.2+ by using the following meta tag: <meta name="viewport" content="width=device-width, user-scalable=no">

Feedback

FIGURE 6.2 – Table de compatibilité CSS du site `www.caniuse.com`

▷ www.caniuse.com
Code web : 264222

▷ www.normansblog.de
Code web : 697092

Où écrit-on le CSS ?

Vous avez le choix car on peut écrire du code en langage CSS à trois endroits différents :

- dans un fichier `.css` (*méthode la plus recommandée*) ;
- dans l'en-tête `<head>` du fichier HTML ;

– directement dans les balises du fichier HTML *via* un attribut style (*méthode la moins recommandée*).

Je vais vous présenter ces trois méthodes mais sachez d'ores et déjà que la première... est la meilleure.

Dans un fichier .css (recommandé)

Comme je viens de vous le dire, on écrit le plus souvent le code CSS dans un fichier spécial ayant l'extension .css (contrairement aux fichiers HTML qui ont l'extension .html). C'est la méthode la plus pratique et la plus souple. Cela nous évite de tout mélanger dans un même fichier. J'utiliserai cette technique dans toute la suite de ce cours.

Commençons à pratiquer dès maintenant ! Nous allons partir du fichier HTML suivant :

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <link rel="stylesheet" href="style.css" />
6          <title>Premiers tests du CSS</title>
7      </head>
8
9      <body>
10         <h1>Mon super site</h1>
11
12         <p>Bonjour et bienvenue sur mon site !</p>
13         <p>Pour le moment, mon site est un peu <em>vide</em>.
              Patientez encore un peu !</p>
14     </body>
15 </html>
```

Vous noterez le contenu de la ligne 5, <link rel="stylesheet" href="style.css" /> : c'est elle qui indique que ce fichier HTML est associé à un fichier appelé style.css et chargé de la mise en forme.

Enregistrez ce fichier sous le nom que vous voulez (par exemple page.html). Pour le moment, rien d'extraordinaire à part la nouvelle balise que nous avons ajoutée.

Maintenant, créez un *nouveau* fichier vide dans votre éditeur de texte (par exemple Notepad++) et copiez-y ce bout de code CSS (rassurez-vous, je vous expliquerai tout à l'heure ce qu'il veut dire) :

```
p
{
    color: blue;
}
```



Pour activer la coloration du code dans Notepad++, passez par les menus `Langage > C > CSS`.

Enregistrez le fichier en lui donnant un nom qui se termine par `.css`, comme `style.css`. Placez ce fichier `.css` dans le même dossier que votre fichier `.html`.

Dans Notepad++, vous devriez observer quelque chose de similaire à la figure 6.3.

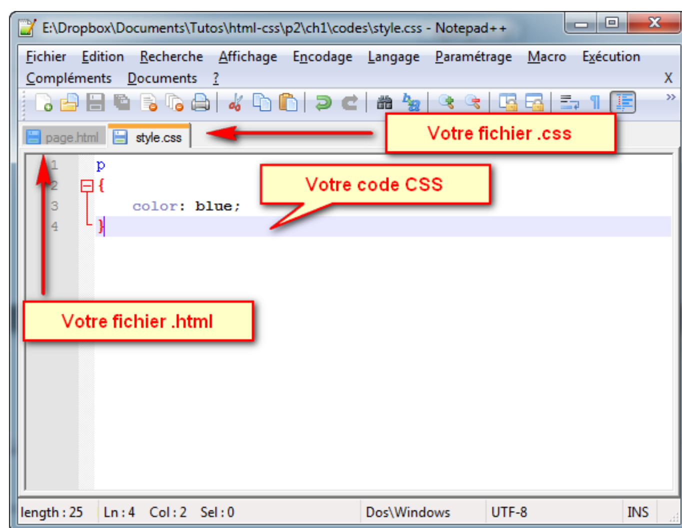


FIGURE 6.3 – Fichiers HTML et CSS dans Notepad++

Dans votre explorateur de fichiers, vous devriez les voir apparaître côte à côte. D'un côté le `.html`, de l'autre le `.css`, comme à la figure 6.4.

Ouvrez maintenant votre fichier `page.html` dans votre navigateur pour le tester, comme vous le faites d'habitude. Regardez, c'est magique : vos paragraphes sont écrits en bleu, comme dans la figure 6.5 !



Il est inutile d'ouvrir directement le fichier `style.css` dans le navigateur. Il faut ouvrir le fichier `page.html` (il fera automatiquement appel au fichier `style.css`).

Dans l'en-tête `<head>` du fichier HTML

Il existe une autre méthode pour utiliser du CSS dans ses fichiers HTML : cela consiste à insérer le code CSS directement dans une balise `<style>` à l'intérieur de l'en-tête `<head>`.

Voici comment on peut obtenir exactement le même résultat avec un seul fichier `.html`

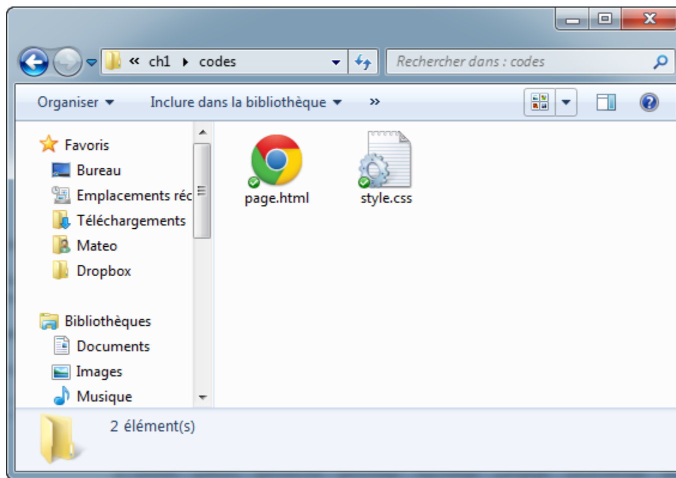


FIGURE 6.4 – Fichiers HTML et CSS dans l'explorateur de fichiers

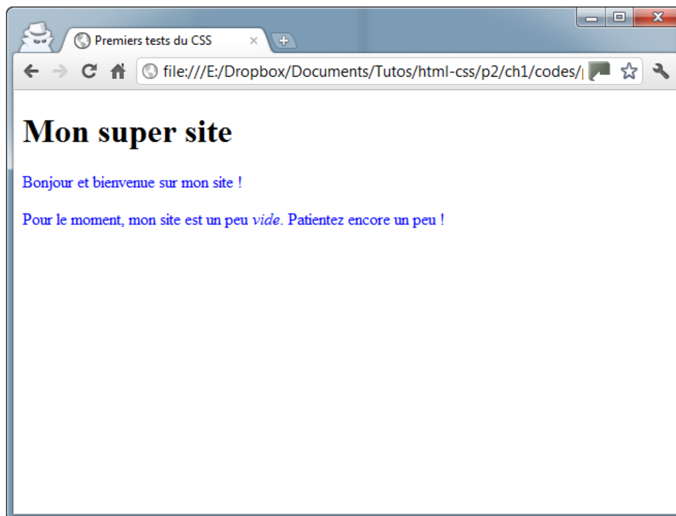


FIGURE 6.5 – Le texte est écrit en bleu

qui contient le code CSS (lignes 5 à 10) :

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <style>
6              p
7              {
8                  color: blue;
9              }
10         </style>
11         <title>Premiers tests du CSS</title>
12     </head>
13
14     <body>
15         <h1>Mon super site</h1>
16
17         <p>Bonjour et bienvenue sur mon site !</p>
18         <p>Pour le moment, mon site est un peu <em>vide</em>. Patientez encore un peu !</p>
19     </body>
20 </html>
```

Testez, vous verrez que le résultat est le même.

Directement dans les balises (non recommandé)

Dernière méthode, à manipuler avec précaution : vous pouvez ajouter un attribut `style` à n'importe quelle balise. Vous insérerez votre code CSS directement dans cet attribut :

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>Premiers tests du CSS</title>
6      </head>
7
8      <body>
9          <h1>Mon super site</h1>
10
11         <p style="color: blue;">Bonjour et bienvenue sur mon site !</p>
12         <p>Pour le moment, mon site est un peu <em>vide</em>. Patientez encore un peu !</p>
13     </body>
14 </html>
```

Cette fois, seule le texte du premier paragraphe (ligne 11), dont la balise contient le code CSS, sera coloré en bleu (figure 6.6).



FIGURE 6.6 – Le premier paragraphe est écrit en bleu

Quelle méthode choisir ?



Je trouve que la première méthode que tu recommandes est plus compliquée que les deux autres ! Pourquoi nous conseilles-tu de créer deux fichiers, j'étais bien, moi, avec juste un fichier `.html` !

Je vous recommande fortement de prendre l'habitude de travailler avec la première méthode parce que c'est celle utilisée par la majorité des webmasters. . . Pourquoi ?

Pour le moment, vous faites vos tests sur un seul fichier HTML. Cependant, votre site sera plus tard constitué de plusieurs pages HTML, on est d'accord ? Imaginez : si vous placez le code CSS directement dans le fichier HTML, il faudra copier ce code dans tous les fichiers HTML de votre site ! Et si demain vous changez d'avis, par exemple si vous voulez que vos paragraphes soient écrits en rouge et non en bleu, il faudra modifier chaque fichier HTML un à un, comme le montre la figure 6.7.

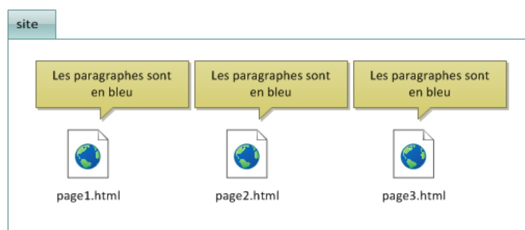


FIGURE 6.7 – Le code CSS est répété dans chaque fichier HTML

Si vous travaillez avec un fichier CSS externe, vous n'aurez besoin d'écrire cette instruction qu'une seule fois pour tout votre site, comme le montre la figure 6.8.

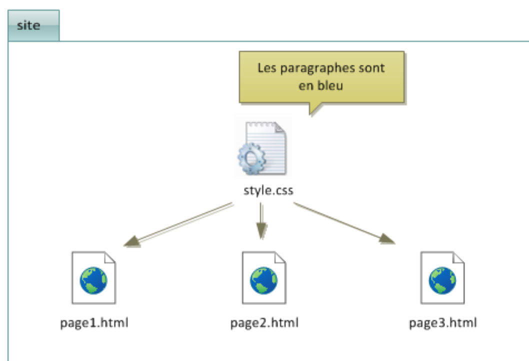


FIGURE 6.8 – Le code CSS est donné une fois pour toutes dans un fichier CSS

Appliquer un style : sélectionner une balise

Maintenant que nous savons où placer le code CSS, intéressons-nous de plus près à ce code. Je vous ai donné, sans vous l'expliquer, un premier bout de code CSS :

```
p
{
    color: blue;
}
```

Dans un code CSS comme celui-ci, on trouve trois éléments différents :

- **Des noms de balises** : on écrit les noms des balises dont on veut modifier l'apparence. Par exemple, si je veux modifier l'apparence de tous les paragraphes `<p>`, je dois écrire `p`.
- **Des propriétés CSS** : les « effets de style » de la page sont rangés dans des propriétés. Il y a par exemple la propriété `color` qui permet d'indiquer la couleur du texte, `font-size` qui permet d'indiquer la taille du texte, etc. Il y a beaucoup de propriétés CSS et, comme je vous l'ai dit, je ne vous obligerai pas à les connaître toutes par cœur.
- **Les valeurs** : pour chaque propriété CSS, on doit indiquer une valeur. Par exemple, pour la propriété `color`, il faut indiquer le nom de la couleur. Pour `font-size`, il faut indiquer quelle taille on veut, etc.

Schématiquement, une feuille de style CSS ressemble donc à cela :

```
balise1
{
    propriete1: valeur1;
    propriete2: valeur2;
```

```
    propriete3: valeur3;
}

balise2
{
    propriete1: valeur1;
    propriete2: valeur2;
    propriete3: valeur3;
    propriete4: valeur4;
}

balise3
{
    propriete1: valeur1;
}
```

Vous repérez dans cet extrait de code les balises, propriétés et valeurs dont je viens de vous parler.

Comme vous le voyez, on écrit le nom de la balise (par exemple `h1`) et on ouvre des accolades pour, à l'intérieur, mettre les propriétés et valeurs que l'on souhaite. On peut mettre autant de propriétés que l'on veut à l'intérieur des accolades. Chaque propriété est suivie du symbole « deux-points » (:) puis de la valeur correspondante. Enfin, chaque ligne se termine par un point-virgule (;).

Je vous apprendrai de nombreuses propriétés dans les chapitres suivants. Pour le moment, dans les exemples, on va juste changer la couleur pour s'entraîner.

Le code CSS que nous avons utilisé jusqu'ici :

```
p
{
    color: blue;
}
```

... signifie donc en français : « *Je veux que tous mes paragraphes soient écrits en bleu.* ». Le résultat est visible à la figure 6.9.

Essayez de changer le nom de la balise affectée par le code CSS. Par exemple, si j'écris `h1`, c'est le titre qui sera écrit en bleu. Modifiez votre fichier `style.css` comme ceci :

```
h1
{
    color: blue;
}
```

Maintenant, ouvrez à nouveau votre page HTML (souvenez-vous, c'est la page HTML qu'on ouvre dans le navigateur, pas le fichier CSS !) : vous devriez voir son titre s'afficher en bleu (figure 6.10) !



Figure 6.9 – Paragraphes écrits en bleu



Figure 6.10 – Titre écrit en bleu

Appliquer un style à plusieurs balises

Prenons le code CSS suivant :

```
h1
{
    color: blue;
}

em
{
    color: blue;
}
```

Il signifie que nos titres <h1> et nos textes importants <em> doivent s'afficher en bleu. Par contre, c'est un peu répétitif, vous ne trouvez pas ?

Heureusement, il existe un moyen en CSS d'aller plus vite si les deux balises doivent avoir la même présentation. Il suffit de combiner la déclaration en séparant les noms des balises par une virgule, comme ceci :

```
h1, em
{
    color: blue;
}
```

Le résultat se trouve à la figure 6.11.

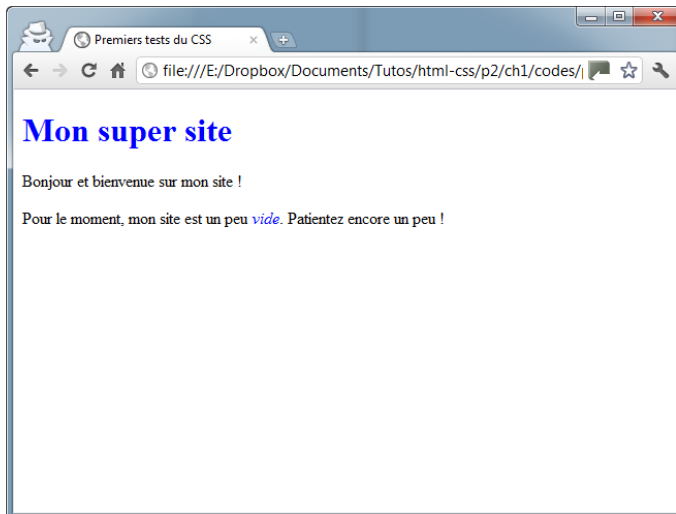


FIGURE 6.11 – Titre et texte important écrits en bleu

Cela signifie : « *Je veux que le texte de mes <h1> et <em> soit écrit en bleu* ».

Vous pouvez indiquer autant de balises à la suite que vous le désirez.

Des commentaires dans du CSS

Comme en HTML, il est possible de mettre des commentaires. Les commentaires ne seront pas affichés, ils servent simplement à indiquer des informations pour vous, par exemple pour vous y retrouver dans un looong fichier CSS.

D'ailleurs, vous allez vous en rendre compte, en général le fichier HTML est assez court et la feuille CSS assez longue (si elle contient tous les éléments de style de votre site, c'est un peu normal). Notez qu'il est possible de créer plusieurs fichiers CSS pour votre site si vous ressentez le besoin de séparer un peu votre code CSS (en fonction des différentes sections de votre site, par exemple).

. . . De quoi on parlait déjà ? Ah oui, les commentaires en CSS.

Donc, pour faire un commentaire, c'est facile ! Tapez `/*`, suivi de votre commentaire, puis `*/` pour terminer votre commentaire. Vos commentaires peuvent être écrits sur une ou plusieurs lignes. Par exemple :

```
/*
style.css
---------

Par Mathieu Nebra
*/

p
{
    color: blue; /* Les paragraphes seront en bleu */
}
```

Il est possible que j'utilise les commentaires dans la suite du cours, pour vous donner des explications à l'intérieur même des fichiers `.css`.

Appliquer un style : `class` et `id`

Ce que je vous ai montré jusqu'ici a quand même un défaut : cela implique par exemple que TOUS les paragraphes possèdent la même présentation (ici, ils seront donc tous écrits en bleu). Comment faire pour que certains paragraphes seulement soient écrits d'une manière différente ? On pourrait placer le code CSS dans un attribut `style` sur la balise que l'on vise (c'est la technique que je vous ai présentée un peu plus tôt) mais, comme je vous l'ai dit, ce n'est pas recommandé (il vaut mieux utiliser un fichier CSS externe).

Pour résoudre le problème, on peut utiliser ces attributs spéciaux *qui fonctionnent sur toutes les balises* :

- l'attribut `class` ;
- l'attribut `id`.

Que les choses soient claires dès le début : les attributs `class` et `id` sont quasiment

identiques. Il y a seulement une petite différence que je vous dévoilerai plus bas. Pour le moment, et pour faire simple, on ne va s'intéresser qu'à l'attribut `class`.

Comme je viens de vous le dire, c'est un attribut que l'on peut mettre sur n'importe quelle balise, aussi bien titre que paragraphe, image, etc.

```
<h1 class=""> </h1>
<p class=""> </p>
<img class="" />
```



Oui mais que met-on comme valeur à l'attribut class ?

En fait, vous devez écrire un nom qui sert à identifier la balise. Ce que vous voulez, du moment que le nom commence par une lettre.

Par exemple, je vais associer la classe `introduction` à mon premier paragraphe (ligne 12) :

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <link rel="stylesheet" href="style.css" />
6          <title>Premiers tests du CSS</title>
7      </head>
8
9      <body>
10         <h1>Mon super site</h1>
11
12         <p class="introduction">Bonjour et bienvenue sur mon
               site !</p>
13         <p>Pour le moment, mon site est un peu <em>vide</em>.
               Patientez encore un peu !</p>
14     </body>
15 </html>
```

Maintenant que c'est fait, votre paragraphe est identifié. Il a un nom : `introduction`. Vous allez pouvoir réutiliser ce nom dans le fichier CSS pour dire : « *Je veux que seules les balises qui ont comme nom 'introduction' soient affichées en bleu* ».

Pour faire cela en CSS, indiquez le nom de votre classe en commençant par un point, comme ci-dessous :

```
.introduction
{
    color: blue;
}
```

Testez le résultat : seul votre paragraphe appelé `introduction` va s'afficher en bleu (figure 6.12) !



FIGURE 6.12 – Seul le premier paragraphe s'affiche en bleu



Et l'attribut `id` alors ?

Lui, il fonctionne exactement de la même manière que `class`, à un détail près : il ne peut être utilisé *qu'une fois* dans le code.

Quel intérêt ? Il y en a assez peu pour tout vous dire, cela vous sera utile si vous faites du JavaScript plus tard pour reconnaître certaines balises. D'ailleurs, nous avons déjà vu l'attribut `id` dans le chapitre sur les liens (pour réaliser des ancres). En pratique, nous ne mettrons des `id` que sur des éléments qui sont uniques dans la page, comme par exemple le logo :

```
<img src="images/logo.png" alt="Logo du site" id="logo" />
```

Si vous utilisez des `id`, lorsque vous définirez leurs propriétés dans le fichier CSS, il faudra faire précéder le nom de l'`id` par un dièse (#) :

```
#logo
{
    /* Indiquez les propriétés CSS ici */
}
```

Je ne vous propose pas de le tester, cela fonctionne exactement comme `class`.



Si vous vous emmêlez les pinceaux entre `class` et `id` retenez que deux balises peuvent avoir le même nom avec l'attribut `class`. Un nom d'`id` doit en revanche être unique dans la page HTML.

Les balises universelles

Il arrivera parfois que vous ayez besoin d'appliquer une `class` (ou un `id`) à certains mots qui, à l'origine, ne sont pas entourés par des balises.

En effet, le problème de `class`, c'est qu'il s'agit d'un attribut. Vous ne pouvez donc en mettre que sur une balise. Si, par exemple, je veux modifier uniquement « bienvenue » dans le paragraphe suivant :

```
<p>Bonjour et bienvenue sur mon site !</p>
```

Cela serait facile à faire s'il y avait une balise autour de « bienvenue » mais, malheureusement il n'y en a pas. Par chance, on a inventé... la balise-qui-ne-sert-à-rien.

En fait, on a inventé deux balises dites **universelles**, qui n'ont aucune signification particulière (elles n'indiquent pas que le mot est important, par exemple). Il y a une différence minime (mais significative !) entre ces deux balises :

- `<span> </span>` : c'est une balise de type **inline**, c'est-à-dire une balise que l'on place au sein d'un paragraphe de texte, pour sélectionner certains mots uniquement. Les balises `<strong>` et `<em>` sont de la même famille. Cette balise s'utilise donc au milieu d'un paragraphe et c'est celle dont nous allons nous servir pour colorer « bienvenue ».
- `<div> </div>` : c'est une balise de type **block**, qui entoure un bloc de texte. Les balises `<p>`, `<h1>`, etc. sont de la même famille. Ces balises ont quelque chose en commun : elles créent un nouveau « bloc » dans la page et provoquent donc obligatoirement un retour à la ligne. `<div>` est une balise fréquemment utilisée dans la construction d'un design, comme nous le verrons plus tard.

Pour le moment donc, nous allons utiliser plutôt la balise `<span>`. On la met autour de « bienvenue », on lui ajoute une classe (du nom qu'on veut), on crée le CSS et c'est gagné !

```
<p>Bonjour et <span class="salutations">bienvenue</span> sur
    mon site !</p>
```

```
.salutations
{
    color: blue;
}
```

Vous pouvez voir le résultat à la figure 6.13.

Appliquer un style : les sélecteurs avancés

En CSS, le plus difficile est de savoir cibler le texte dont on veut changer la forme. Pour cibler (on dit « sélectionner ») les éléments de la page à modifier, on utilise ce qu'on appelle des **sélecteurs**. Vous en avez déjà utilisé quelques-uns un peu plus tôt dans ce chapitre, résumons-les pour commencer.



FIGURE 6.13 – Le mot « bienvenue » est écrit en bleu

Les sélecteurs que vous connaissez déjà

Ces sélecteurs, que nous avons vus précédemment, sont de loin les plus couramment utilisés. Il faut les connaître par cœur. Commençons par la base de la base :

```
p
{

}
```

... signifie « Je veux toucher tous les paragraphes ». Après, c'est à vous de dire ce que vous faites à ces paragraphes (vous les écrivez en bleu, par exemple).

Nous avons aussi vu :

```
h1 , em
{

}
```

... qui signifie « Tous les titres et tous les textes importants ». Nous avons sélectionné deux balises d'un coup.

Et enfin, nous avons vu comment sélectionner des balises précises à qui nous avons donné un nom grâce aux attributs `class` et `id` :

```
.class
{

}

```

```
#id
{

}
```

Vous savez quoi ? Il existe des dizaines d'autres façons de cibler des balises en CSS ! Nous n'allons pas toutes les voir car il y en a beaucoup et certaines sont complexes, mais voici déjà de quoi vous permettre d'être plus efficaces en CSS !

Les sélecteurs avancés

* : sélecteur universel

```
*
{

}
```

Sélectionne toutes les balises sans exception. On l'appelle le sélecteur universel.

A B : une balise contenue dans une autre

```
h3 em
{

}
```

Sélectionne toutes les balises `<em>` situées à l'intérieur d'une balise `<h3>`. Notez qu'il n'y a pas de virgule entre les deux noms de balises.

Exemple de code HTML correspondant :

```
<h3>Titre avec <em>texte important</em></h3>
```

A + B : une balise qui en suit une autre

```
h3 + p
{

}
```

Sélectionne la première balise `<p>` située après un titre `<h3>`.

Exemple :

```
<h3>Titre</h3>

<p>Paragraphe</p>
```

A[attribut] : une balise qui possède un attribut

```
a[title]
{

}
```

Sélectionne tous les liens `<a>` qui possèdent un attribut `title`.

Exemple :

```
<a href="http://site.com" title="Infobulle">
```

A[attribut="Valeur"] : une balise, un attribut et une valeur exacte

```
a[title="Cliquez ici"]
{

}
```

Idem, mais l'attribut doit en plus avoir exactement pour valeur « Cliquez ici ».

Exemple :

```
<a href="http://site.com" title="Cliquez ici">
```

A[attribut*="Valeur"] : une balise, un attribut et une valeur

```
a[title*="ici"]
{

}
```

Idem, l'attribut doit cette fois contenir dans sa valeur le mot « ici » (peu importe sa position).

Exemple :

```
<a href="http://site.com" title="Quelque part par ici">
```

D'autres sélecteurs existent !

Je ne vous ai présenté ici qu'une partie des sélecteurs CSS mais sachez qu'il en existe beaucoup d'autres. Si vous voulez une liste complète, vous pouvez vous renseigner directement à la source : sur le site du W3C ! C'est très complet.

▷ Liste des sélecteurs
Code web : 733661

Sachez que nous découvrirons certains de ces autres sélecteurs dans la suite de ce cours !

En résumé

– CSS est un autre langage qui vient compléter le HTML. Son rôle est de mettre en forme votre page web.

- Il faut être vigilant sur la compatibilité des navigateurs avec certaines fonctionnalités récentes de CSS3. Quand un navigateur ne connaît pas une instruction de mise en forme, il l'ignore simplement.
- On peut écrire le code CSS à plusieurs endroits différents, le plus conseillé étant de créer un fichier séparé portant l'extension `.css` (exemple : `style.css`).
- En CSS, on sélectionne quelles portions de la page HTML on veut modifier et on change leur présentation avec des propriétés CSS :

```
balise1
{
    propriete1: valeur1;
    propriete2: valeur2;
}
```

- Il existe de nombreuses façons de sélectionner la portion de la page que l'on veut mettre en forme. Par exemple, on peut viser :
  - toutes les balises d'un même type, en écrivant simplement leur nom (`h1` par exemple) ;
  - certaines balises spécifiques, auxquelles on a donné des noms à l'aide des attributs `class` ou `id` (`.nomclasse` ou `#nomid`) ;
  - uniquement les balises qui se trouvent à l'intérieur d'autres balises (`h3 em`).
  - etc.

Chapitre 7

Formatage du texte

Difficulté :

Nous arrivons maintenant à un chapitre qui devrait beaucoup vous intéresser. Non, le « formatage du texte » n'a rien à voir avec la destruction de toutes les données présentes sur votre disque dur ! Cela signifie simplement que l'on va modifier l'apparence du texte (on dit qu'on le « met en forme »).

Pas de surprise particulière : nous sommes toujours dans le CSS et nous allons réutiliser ce que nous venons d'apprendre dans le chapitre précédent. Nous allons donc travailler directement au sein du fichier `.css` que nous avons créé.

Ce chapitre va être l'occasion de découvrir de nombreuses propriétés CSS : nous allons voir comment modifier la taille du texte, changer la police, aligner le texte...



La taille

Pour modifier la taille du texte, on utilise la propriété CSS `font-size`. Mais comment indiquer la taille du texte ? C'est là que les choses se corsent car plusieurs techniques vous sont proposées :

- Indiquer une **taille absolue** : en pixels, en centimètres ou millimètres. Cette méthode est très précise mais il est conseillé de ne l'utiliser que si c'est absolument nécessaire, car on risque d'indiquer une taille trop petite pour certains lecteurs.
- Indiquer une **taille relative** : en pourcentage, « em » ou « ex », cette technique a l'avantage d'être plus souple. Elle s'adapte plus facilement aux préférences de taille des visiteurs.

Une taille absolue

Pour indiquer une taille absolue, on utilise généralement les pixels. Pour avoir un texte de 16 pixels de hauteur, vous devez donc écrire :

```
font-size: 16px;
```

Les lettres auront une taille de 16 pixels, comme le montre la figure 7.1.

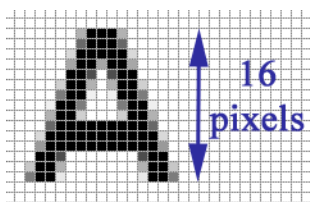


FIGURE 7.1 – Une lettre de 16 pixels de hauteur

Voici un exemple d'utilisation (placez ce code dans votre fichier `.css`) :

```
p
{
    font-size: 14px; /* Paragraphes de 14 pixels */
}
h1
{
    font-size: 40px; /* Titres de 40 pixels */
}
```

Et le résultat est visible à la figure 7.2.



Si vous le souhaitez, vous pouvez également définir des tailles en centimètres ou millimètres. Remplacez « px » par « cm » ou « mm ». Ces unités sont cependant moins bien adaptées aux écrans.



FIGURE 7.2 – Différentes tailles de texte

Une valeur relative

C'est la méthode recommandée car le texte s'adapte alors plus facilement aux préférences de tous les visiteurs.

Il y a plusieurs moyens d'indiquer une valeur relative. Vous pouvez par exemple écrire la taille avec des mots en anglais comme ceux-ci :

- `xx-small` : minuscule ;
- `x-small` : très petit ;
- `small` : petit ;
- `medium` : moyen ;
- `large` : grand ;
- `x-large` : très grand ;
- `xx-large` : euh... gigantesque.

Vous pouvez tester l'utilisation de ces valeurs dans votre code CSS :

```
p
{
    font-size: small;
}
h1
{
    font-size: large;
}
```

Bon, cette technique a un défaut : il n'y a que sept tailles disponibles (car il n'y a que sept noms). Heureusement, il existe d'autres moyens. Ma technique préférée consiste à indiquer la taille en « em ».

- Si vous écrivez `1em`, le texte a une taille normale.
- Si vous voulez grossir le texte, vous pouvez inscrire une valeur supérieure à 1, comme `1.3em`.
- Si vous voulez réduire le texte, inscrivez une valeur inférieure à 1, comme `0.8em`.



Faites attention : pour les nombres décimaux, il faut mettre un point et non une virgule. Vous devez donc écrire « 1.4em » et non pas « 1,4em » !

Exemple :

```
p
{
    font-size: 0.8em;
}
h1
{
    font-size: 1.3em;
}
```

D'autres unités sont disponibles. Vous pouvez essayer le « ex » (qui fonctionne sur le même principe que le em mais qui est plus petit de base) et le pourcentage (80%, 130%...).

La police

Ah... La police... On touche un point sensible.

En effet, il se pose un problème : pour qu'une police s'affiche correctement, il faut que tous les internautes l'aient. Si un internaute n'a pas la même police que vous, son navigateur prendra une police par défaut (une police standard) qui n'aura peut-être rien à voir avec ce à quoi vous vous attendiez.

La bonne nouvelle, c'est que depuis CSS 3, il est possible de faire télécharger automatiquement une police au navigateur. Je vous expliquerai dans un second temps comment faire cela.

Modifier la police utilisée

La propriété CSS qui permet d'indiquer la police à utiliser est `font-family`. Vous devez écrire le nom de la police comme ceci :

```
balise
{
    font-family: police;
}
```

Seulement, pour éviter les problèmes si l'internaute n'a pas la même police que vous, on précise en général *plusieurs* noms de police, séparés par des virgules :

```
balise
{
    font-family: police1, police2, police3, police4;
}
```

Le navigateur essaiera d'abord d'utiliser la `police1`. S'il ne l'a pas, il essaiera la `police2`. S'il ne l'a pas, il passera à la `police3`, et ainsi de suite. En général, on indique en tout dernier `serif`, ce qui correspond à une police par défaut (qui ne s'applique que si aucune autre police n'a été trouvée).



Il existe aussi une autre police par défaut appelée `sans-serif`. La différence entre les deux est la présence de petites pattes de liaison en bas des lettres, que la police `sans-serif` n'a pas. Oui, c'est subtil.

Oui, mais quelles sont les polices les plus courantes qu'on a le « droit » d'utiliser, me direz-vous ? Voici une liste de polices qui fonctionnent bien sur la plupart des navigateurs :

- Arial ;
- Arial Black ;
- Comic Sans MS ;
- Courier New ;
- Georgia ;
- Impact ;
- Times New Roman ;
- Trebuchet MS ;
- Verdana.

La figure 7.3 vous montre à quoi ressemblent ces polices.

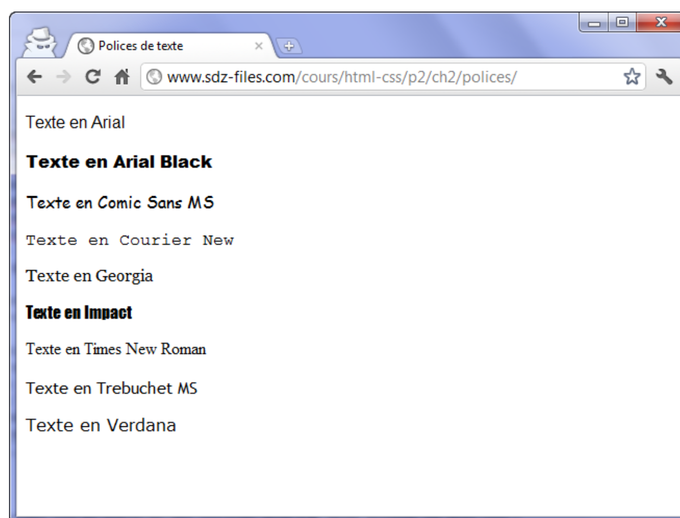


FIGURE 7.3 – Différentes polices

Ainsi, si j'écris :

```
p
{
```

```
    font-family: Impact, "Arial Black", Arial, Verdana, serif;
}
```

... cela signifie : « *Mets la police Impact ou, si elle n'y est pas, Arial Black, ou sinon Arial, ou sinon Verdana, ou si rien n'a marché, mets une police standard (serif)* ».

En général, il est bien d'indiquer un choix de trois ou quatre polices (+ serif) afin de s'assurer qu'au moins l'une d'entre elles aura été trouvée sur l'ordinateur du visiteur.



Si le nom de la police comporte des espaces, je conseille de l'entourer de guillemets, comme je l'ai fait pour « Arial Black ».

Utiliser une police personnalisée avec `@font-face`



Je trouve le choix des polices trop limité. Comment puis-je utiliser ma police préférée sur mon site web ?

Pendant longtemps, cela n'était pas possible. Aujourd'hui, avec CSS 3, il existe heureusement un moyen d'utiliser n'importe quelle police sur son site. Cela fonctionne bien avec la plupart des navigateurs.

Mais attention, il y a des défauts (ce serait trop beau sinon) :

- Il faudra que le navigateur de vos visiteurs *télécharge* automatiquement le fichier de la police, dont le poids peut atteindre, voire dépasser 1 Mo...
- La plupart des polices sont soumises au droit d'auteur, il n'est donc *pas légal* de les utiliser sur son site. Heureusement, il existe des sites comme `www.fontsquirrel.com` et `www.dafont.com` qui proposent en téléchargement un certain nombre de polices libres de droits. Je recommande en particulier `www.fontsquirrel.com` car il permet de télécharger des packs prêts à l'emploi pour CSS 3.
- Il existe *plusieurs formats* de fichiers de polices et ceux-ci ne fonctionnent pas sur tous les navigateurs.

Voici les différents formats de fichiers de polices qui existent et qu'il faut connaître :

- `.ttf` : *TrueType Font*. Fonctionne sur IE9 et tous les autres navigateurs.
- `.eot` : *Embedded OpenType*. Fonctionne sur Internet Explorer uniquement, toutes versions. Ce format est propriétaire, produit par Microsoft.
- `.otf` : *OpenType Font*. Ne fonctionne pas sur Internet Explorer.
- `.svg` : *SVG Font*. Le seul format reconnu sur les iPhones et iPads pour le moment.
- `.woff` : *Web Open Font Format*. Nouveau format conçu pour le Web, qui fonctionne sur IE9 et tous les autres navigateurs.

En CSS, pour définir une nouvelle police, vous devez la déclarer comme ceci :

```
@font-face {
    font-family: 'MaSuperPolice';
```

```
    src: url('MaSuperPolice.eot');
}
```

Le fichier de police (ici *MaSuperPolice.eot*) doit ici être situé dans le même dossier que le fichier CSS (ou dans un sous-dossier, si vous utilisez un chemin relatif).



Je croyais qu'il y avait plusieurs formats de police ?

Oui, d'ailleurs les `.eot` ne fonctionnent que sur Internet Explorer. L'idéal est de proposer plusieurs formats pour la police : le navigateur téléchargera celui qu'il sait lire. Voici comment indiquer plusieurs formats :

```
@font-face {
    font-family: 'MaSuperPolice';
    src: url('MaSuperPolice.eot') format('eot'),
         url('MaSuperPolice.woff') format('woff'),
         url('MaSuperPolice.ttf') format('truetype'),
         url('MaSuperPolice.svg') format('svg');
}
```

Pour tester le fonctionnement, je vous propose de télécharger une police sur Font Squirrel, par exemple `Learning Curve Pro`. Cliquez sur « @font-face Kit », cela vous permettra de télécharger un kit prêt à l'emploi avec tous les formats pour cette police.

▷ Télécharger Learning Curve Pro
Code web : 771058

Votre fichier CSS ressemblera au final à ceci :

```
@font-face { /* Définition d'une nouvelle police nommée
    LearningCurveProRegular */
    font-family: 'LearningCurveProRegular';
    src: url('LearningCurve_OT-webfont.eot');
    src: url('LearningCurve_OT-webfont.eot?#iefix') format('
        embedded-opentype'),
         url('LearningCurve_OT-webfont.woff') format('woff'),
         url('LearningCurve_OT-webfont.ttf') format('truetype')
            ,
         url('LearningCurve_OT-webfont.svg#
            LearningCurveProRegular') format('svg');
}

h1 /* Utilisation de la police qu'on vient de définir sur les
    titres */
{
    font-family: 'LearningCurveProRegular', Arial, serif;
}
```

La première (grosse) section `@font-face` permet de définir un nouveau nom de police qui pourra être utilisé dans le fichier CSS. Ensuite, nous utilisons ce nom de police avec la propriété `font-family`, que nous connaissons, pour modifier l'apparence des titres `<h1>`. Vous pouvez voir le résultat à la figure 7.4.

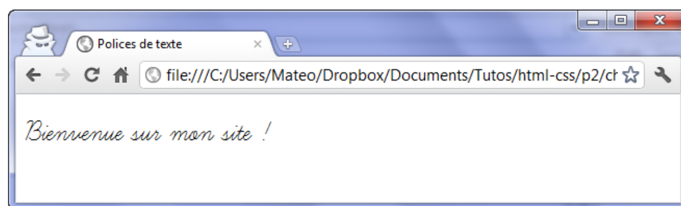


Figure 7.4 – Affichage d'une police personnalisée



Vous noterez quelques bizarreries dans le CSS généré par le site Font Squirrel. Le but est de pallier certains bugs sur Internet Explorer car les anciennes versions ne comprennent pas quand on définit plusieurs formats. Cela explique donc la présence d'un `?#iefix` dans le code.

Italique, gras, souligné...

Il existe en CSS une série de propriétés classiques de mise en forme du texte. Nous allons découvrir ici comment afficher le texte en gras, italique, souligné... et au passage nous verrons qu'il est même possible d'aller jusqu'à le faire clignoter !

Mettre en italique



Attends un peu là ! Je croyais que la balise `<em>` permettait de mettre un texte en italique ? !

Je n'ai jamais dit cela. Retournez voir les chapitres précédents si vous avez des doutes, mais je n'ai *jamais* dit que la balise `<em>` était faite pour mettre le texte en italique (de même que je n'ai jamais dit que `<strong>` était faite pour mettre en gras).

`<em>`, mettez-vous bien cela dans la tête, est faite pour insister sur des mots. Cela veut dire que les mots qu'elle entoure sont assez importants. Pour représenter cette importance, la plupart des navigateurs choisissent d'afficher le texte en italique, mais ce n'est pas une obligation.

Le CSS lui, permet de dire réellement : « Je veux que ce texte soit en italique ». Rien ne vous empêche, par exemple, de décider que tous vos titres seront en italique.

Concrètement, en CSS, pour mettre en italique, on utilise `font-style` qui peut prendre trois valeurs :

- `italic` : le texte sera mis en italique.
- `oblique` : le texte sera passé en oblique (les lettres sont penchées, le résultat est légèrement différent de l'italique proprement dit).
- `normal` : le texte sera normal (par défaut). Cela vous permet d'annuler une mise en italique. Par exemple, si vous voulez que les textes entre `<em>` ne soient plus en italique, vous devrez écrire :

```
em
{
    font-style: normal;
}
```

Ainsi, dans l'exemple suivant, je me sers de `font-style` pour mettre en italique tous mes titres `<h2>` :

```
h2
{
    font-style: italic;
}
```

Mettre en gras

Et si nous passions à la mise en gras ? Alors, là encore, n'oubliez pas que ce n'est pas `<strong>` qui permet de mettre en gras (son rôle est d'indiquer que le texte est important, *donc* le navigateur l'affiche généralement en gras). La mise en gras en CSS peut par exemple s'appliquer aux titres, à certains paragraphes entiers, etc. C'est à vous de voir.

La propriété CSS pour mettre en gras est `font-weight` et prend les valeurs suivantes :

- `bold` : le texte sera en gras ;
- `normal` : le texte sera écrit normalement (par défaut).

Voici par exemple comment écrire les titres en gras :

```
h1
{
    font-weight: bold;
}
```

Soulignement et autres décorations

La propriété CSS associée porte bien son nom : `text-decoration`. Elle permet, entre autres, de souligner le texte, mais pas seulement. Voici les différentes valeurs qu'elle peut prendre :

- `underline` : souligné.

- `line-through` : barré.
- `overline` : ligne au-dessus.
- `blink` : clignotant. Ne fonctionne pas sur tous les navigateurs (Internet Explorer et Google Chrome, notamment).
- `none` : normal (par défaut).

Ce CSS va vous permettre de tester les effets de `text-decoration` :

```
h1
{
    text-decoration: blink;
}
.souligne
{
    text-decoration: underline;
}
.barre
{
    text-decoration: line-through;
}
.ligne_dessus
{
    text-decoration: overline;
}
```

Et le résultat est visible à la figure 7.5.



FIGURE 7.5 – Différentes mises en forme du texte

L'alignement

Le langage CSS nous permet de faire tous les alignements connus : à gauche, centré, à droite et justifié.

C'est tout simple. On utilise la propriété `text-align` et on indique l'alignement désiré :

- `left` : le texte sera aligné à gauche (c'est le réglage par défaut).
- `center` : le texte sera centré.
- `right` : le texte sera aligné à droite.

– `justify` : le texte sera « justifié ». Justifier le texte permet de faire en sorte qu'il prenne toute la largeur possible sans laisser d'espace blanc à la fin des lignes. Les textes des journaux, par exemple, sont toujours justifiés.

Regardez les différents alignements sur cet exemple :

```
h1
{
    text-align: center;
}

p
{
    text-align: justify;
}

.signature
{
    text-align: right;
}
```

Le résultat est visible à la figure 7.6.



FIGURE 7.6 – Alignements du texte



Vous ne pouvez pas modifier l'alignement du texte d'une balise *inline* (comme `<span>`, `<a>`, `<em>`, `<strong>`...). L'alignement ne fonctionne que sur des balises de type *block* (`<p>`, `<div>`, `<h1>`, `<h2>`, ...) et c'est un peu logique, quand on y pense : on ne peut pas modifier l'alignement de quelques mots au milieu d'un paragraphe ! C'est donc en général le paragraphe entier qu'il vous faudra aligner.

Les flottants

Le CSS nous permet de faire flotter un élément autour du texte. On dit aussi qu'on fait un « habillage ».

Pour que vous voyiez bien de quoi on parle, la figure 13.1 vous montre ce que nous allons apprendre à faire.

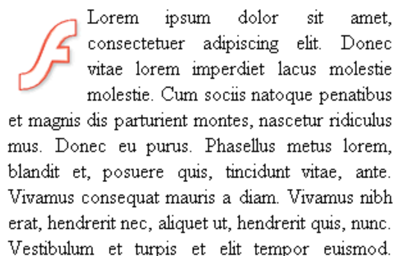


FIGURE 7.7 – Une image flottante entourée par du texte

J'imagine que, maintenant, la question qui vous brûle les lèvres est : « *Mais quelle est donc la propriété magique qui fait flotter ?* ». La réponse est... `float` (« flottant » en anglais). Cette propriété peut prendre deux valeurs très simples :

- `left` : l'élément flottera à gauche.
- `right` : l'élément flottera... à droite ! Oui, bravo.

L'utilisation des flottants est très simple :

1. Vous appliquez un `float` à une balise.
2. Puis vous continuez à écrire du texte à la suite normalement.



On peut aussi bien utiliser la propriété `float` sur des balises *block* que sur des balises *inline*. Il est courant de faire flotter une image pour qu'elle soit habillée par du texte, comme dans l'exemple précédent.

Faire flotter une image

Nous allons apprendre ici à faire flotter une image. Voici le code HTML que nous devons taper dans un premier temps :

```
<p><img src="flash.gif" class="imageflottante" alt="Image
    flottante" /> Ceci est un texte normal de paragraphe, écrit
    à la suite de l'image et qui l'habillera car l'image est
    flottante.</p>
```



Vous devez placer l'élément flottant en premier dans le code HTML. Si vous placez l'image après le paragraphe, l'effet ne fonctionnera pas.

Voici le seul bout de code CSS qu'on ait besoin de taper, qui permet de faire flotter l'image à gauche :

```
.imageflottante
{
    float: left;
}
```

Amusez-vous aussi à faire flotter l'image à droite, c'est tout bête : il suffit d'indiquer la valeur `right` et le tour est joué !

Stopper un flottant

Quand vous mettez en place un flottant, le texte autour l'habille. Mais comment faire si vous voulez qu'au bout d'un moment le texte continue *en dessous* du flottant ? On pourrait enchaîner plusieurs `<br />` à la suite mais cela ne serait ni élégant ni très propre...

En gros, on aimerait pouvoir obtenir le même résultat qu'à la figure 7.8.

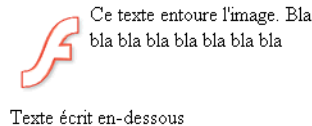


FIGURE 7.8 – Le texte sous l'image ignore la propriété `float`

Il existe en fait une propriété CSS qui permet de dire : « *Stop, ce texte doit être en-dessous du flottant et non plus à côté* ». C'est la propriété `clear`, qui peut prendre ces trois valeurs :

- `left` : le texte se poursuit en-dessous après un `float: left;`
- `right` : le texte se poursuit en-dessous après un `float: right;`
- `both` : le texte se poursuit en-dessous, que ce soit après un `float: left;` ou après un `float: right;`.

Pour simplifier, on va utiliser tout le temps le `clear: both`, qui marche après un flottant à gauche et après un flottant à droite (cela fonctionne donc à tous les coups). Pour illustrer son fonctionnement, on va prendre ce code HTML :

```
<p><img src="flash.gif" class="imageflottante" alt="Image
    flottante" /></p>
<p>Ce texte est écrit à côté de l'image flottante.</p>
<p class="dessous">Ce texte est écrit sous l'image flottante.</
    p>
```

Et ce code CSS :

```
.imageflottante
{
    float: left;
}
.dessous
{
    clear: both;
}
```

Et voilà le travail.

On applique un `clear: both;` au paragraphe que l'on veut voir continuer sous l'image flottante et le tour est joué !

En résumé

- On modifie la taille du texte avec la propriété CSS `font-size`. On peut indiquer la taille en pixels (16px), en « em » (1.3em), en pourcentage (110%), etc.
- On change la police du texte avec `font-family`. Attention, seules quelques polices sont connues par tous les ordinateurs. Vous pouvez cependant utiliser une police personnalisée avec la directive `@font-face` : cela forcera les navigateurs à télécharger la police de votre choix.
- De nombreuses propriétés de mise en forme du texte existent : `font-style` pour l'italique, `font-weight` pour la mise en gras, `text-decoration` pour le soulignement, etc.
- Le texte peut être aligné avec `text-align`.
- On peut faire en sorte qu'une image soit habillée (« entourée ») de texte avec `float`.

Chapitre 8

La couleur et le fond

Difficulté :

Continuons notre tour d'horizon des propriétés CSS existantes. Nous allons nous intéresser ici aux propriétés liées de près ou de loin à la couleur. Nous verrons entre autres :

- comment changer la couleur du texte ;
- comment mettre une couleur ou une image d'arrière-plan ;
- comment ajouter des ombres ;
- comment jouer avec les niveaux de transparence.

Le CSS n'a pas fini de nous étonner !



Couleur du texte

Passons maintenant au vaste sujet de la couleur.



Comment cela, « vaste » ?

Vous connaissez déjà la propriété qui permet de modifier la couleur du texte : il s'agit de `color`. Nous allons nous intéresser aux différentes façons d'indiquer la couleur, car il y en a plusieurs.

Indiquer le nom de la couleur

La méthode la plus simple et la plus pratique pour choisir une couleur consiste à taper son nom (*in english, of course*). Le seul défaut de cette méthode est qu'il n'existe que seize couleurs dites « standard ». D'autres couleurs officieuses existent mais, comme elles ne fonctionneront pas forcément de la même manière sur tous les navigateurs, je vais éviter de vous les montrer.

La figure 8.1 vous montre les seize couleurs que vous pouvez utiliser en tapant simplement leur nom.

Vous pouvez les apprendre par cœur si cela vous chante, en plus cela vous fera réviser votre anglais.

Pour passer tous les titres en marron, on peut donc écrire :

```
h1
{
    color: maroon;
}
```

Vous trouverez le résultat à la figure 8.2.

La notation hexadécimale

Seize couleurs, c'est quand même un peu limite quand on sait que la plupart des écrans peuvent en afficher seize *millions*. D'un autre côté, remarquez, s'il avait fallu donner un nom à chacune des seize millions de couleurs...

Heureusement, il existe en CSS plusieurs façons de choisir une couleur parmi toutes celles qui existent. La première que je vais vous montrer est la notation hexadécimale. Elle est couramment utilisée sur le Web mais il existe aussi une autre méthode que nous verrons plus loin.

Un nom de couleur en hexadécimal, cela ressemble à : #FF5A28. Pour faire simple, c'est une combinaison de lettres et de chiffres qui indiquent une couleur. On doit toujours commencer par écrire un dièse (#), suivi de six lettres ou chiffres allant de 0 à 9

white	
silver	
grey	
black	
red	
maroon	
lime	
green	
yellow	
olive	
blue	
navy	
fuchsia	
purple	
aqua	
teal	

Figure 8.1 – Les seize noms de couleurs utilisables en CSS



Figure 8.2 – Le titre est écrit en marron

et de A à F. Ces lettres ou chiffres fonctionnent deux par deux. Les deux premiers indiquent une quantité de rouge, les deux suivants une quantité de vert et les deux derniers une quantité de bleu. En mélangeant ces quantités (qui sont les composantes Rouge-Vert-Bleu de la couleur) on peut obtenir la couleur qu'on veut.

Ainsi, #000000 correspond à la couleur noire et #FFFFFF à la couleur blanche. Mais maintenant, ne me demandez pas quelle est la combinaison qui produit de l'orange couleur « coucher de soleil », je n'en sais strictement rien.



Certains logiciels de dessin, comme Photoshop, Gimp et Paint.NET, vous indiquent les couleurs en hexadécimal. Il vous est alors facile de copier-coller le code hexadécimal d'une couleur dans votre fichier CSS.

Voici par exemple comment on fait pour appliquer aux paragraphes la couleur blanche en hexadécimal :

```
p
{
    color: #FFFFFF;
}
```



Notez qu'il existe une notation raccourcie : on peut écrire une couleur avec seulement trois caractères. Par exemple : #FA3 équivaut à écrire #FFAA33.

La méthode RGB

Que signifie RGB ? En anglais, Rouge-Vert-Bleu s'écrit *Red-Green-Blue*, ce qui s'abrège en « RGB ». Comme avec la notation hexadécimale, pour choisir une couleur, on doit définir une quantité de rouge, de vert et de bleu.



Encore cette histoire tordue de quantités de rouge-vert-bleu ?

Oui mais là, vous allez voir que c'est beaucoup plus pratique et qu'avec un logiciel de dessin tout simple comme Paint, vous pouvez trouver la couleur que vous désirez. Voici la marche à suivre :

1. Lancez le logiciel Paint depuis le menu `Démarrer`.
2. Rendez-vous dans la section `Modifier les couleurs`, comme indiqué à la figure 8.3.
3. Une fenêtre s'ouvre. Dans la zone qui apparaît à droite, faites bouger les curseurs pour sélectionner la couleur qui vous intéresse. Supposons que vous soyez pris d'une envie folle d'écrire vos titres `<h1>` en rose bonbon (supposons seulement). Sélectionnez la couleur dans la fenêtre, comme à la figure 8.4.

4. Relevez les quantités de Rouge-Vert-Bleu correspondantes, indiquées en bas à droite de la fenêtre (ici 240-96-204). Recopiez ces valeurs dans cet ordre dans le fichier CSS, comme dans le code ci-dessous.

```
p
{
    color: rgb(240,96,204);
}
```

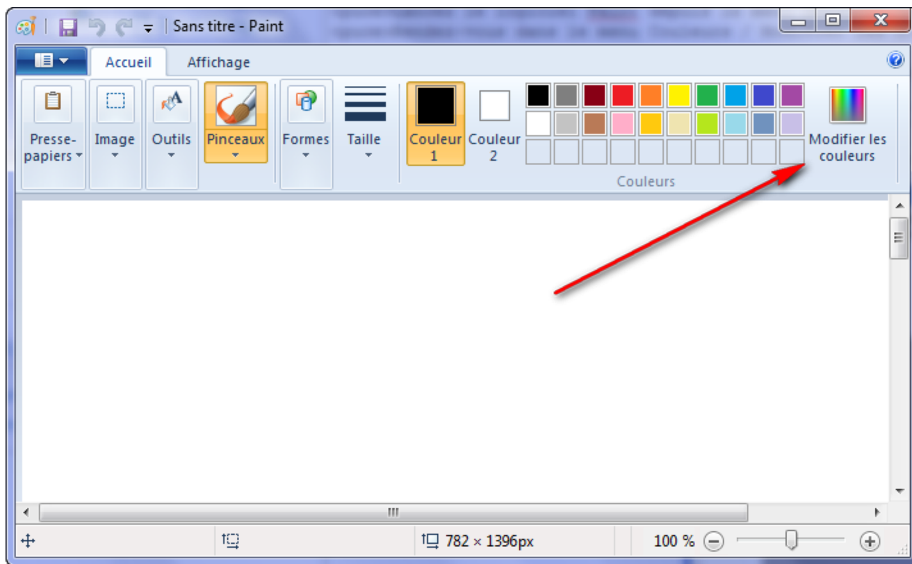


FIGURE 8.3 – Modification des couleurs sous Paint

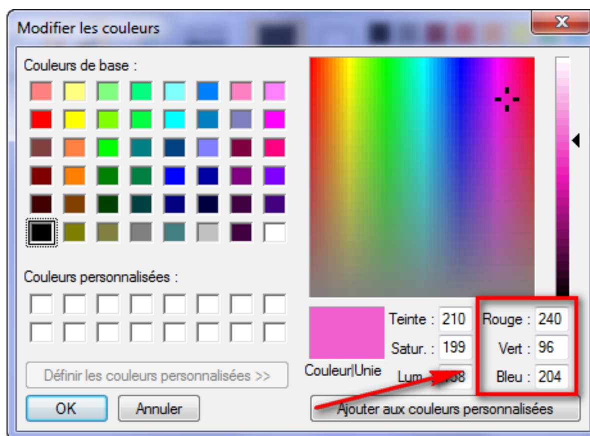


FIGURE 8.4 – Sélection d'une couleur dans Paint

Comme vous avez pu le constater dans l'exemple, pour utiliser la méthode RGB, il faut taper `rgb(Rouge, Vert, Bleu)` en remplaçant « Rouge, Vert, Bleu » par les nombres correspondants. Pour information, ces quantités sont toujours comprises entre 0 et 255.

Et en Bonus Track...

Je vous conseille un petit logiciel tout simple, gratuit et en français, réalisé par Benjamin Chartier, précisément conçu pour choisir vos couleurs. Nul doute qu'il vous sera très utile pour vous aider à choisir vos couleurs. Ce logiciel s'appelle « La boîte à couleurs » (figure 8.5).

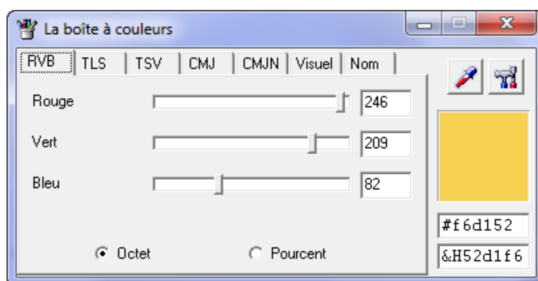


FIGURE 8.5 – La boîte à couleurs

▷ Télécharger le logiciel
Code web : 414953

Comme vous pouvez le voir, il comporte plusieurs onglets. Je vous recommande de rester sur le premier (`RVB`) ou d'aller dans l'onglet `Visuel`. Les autres ne nous concernent pas. Évitez en particulier l'onglet `Nom`, qui propose des noms de couleurs parfois invalides (je vous rappelle qu'il existe seulement seize noms de couleurs « standard »). Vous pouvez récupérer en bas à droite le numéro de la couleur en hexadécimal (le numéro commence toujours par un #) ou encore recopier les valeurs de Rouge-Vert-Bleu (RVB) dans le CSS.

Enfin, et c'est certainement la fonctionnalité la plus intéressante du logiciel, vous pouvez utiliser la pipette en haut à droite pour récupérer n'importe quelle couleur s'affichant sur votre écran ! Amusez-vous bien !

Couleur de fond

Pour indiquer une couleur de fond, on utilise la propriété CSS `background-color`. Elle s'utilise de la même manière que la propriété `color`, c'est-à-dire que vous pouvez taper le nom d'une couleur, l'écrire en notation hexadécimale ou encore utiliser la méthode RGB.

Pour indiquer la couleur de fond de la page web, il faut travailler sur la balise `<body>`. Eh oui, `<body>` correspond à l'ensemble de la page web, c'est donc en modifiant sa

couleur de fond que l'on changera la couleur d'arrière-plan de la page.

Regardez très attentivement ce fichier CSS :

```
/* On travaille sur la balise body, donc sur TOUTE la page */
body
{
    background-color: black; /* Le fond de la page sera noir */
    color: white; /* Le texte de la page sera blanc */
}
```

Voici le rendu de ce code en figure 8.6.



FIGURE 8.6 – Texte en blanc sur fond noir



Eh mais tu as demandé à ce que le texte de la balise <body> soit écrit en blanc, et tous les paragraphes <p> et titres <h1> ont pris cette couleur. Comment cela se fait-il ?

Je voulais justement profiter de l'occasion pour vous en parler. Ce phénomène s'appelle **l'héritage**. Je vous rassure tout de suite, personne n'est mort.

Le CSS et l'héritage

En CSS, si vous appliquez un style à une balise, toutes les balises qui se trouvent à l'intérieur prendront le même style.

C'est en fait simple à comprendre et intuitif. La balise <body>, vous le savez, contient entre autres les balises de paragraphe <p> et de titre <h1>.

Si j'applique une couleur de fond noire et une couleur de texte blanche à la balise <body>, tous mes titres et paragraphes auront eux aussi un arrière-plan de couleur noire et un texte de couleur blanche... C'est ce phénomène qu'on appelle l'héritage : on dit que les balises qui se trouvent à l'intérieur d'une autre balise « héritent » de ses propriétés.



C'est d'ailleurs de là que vient le nom « CSS », qui signifie « *Cascading Style Sheets* », c'est-à-dire « Feuilles de style en cascade ». Les propriétés CSS sont héritées en cascade : si vous donnez un style à un élément, tous les sous-éléments auront le même style.



Cela veut dire que TOUT le texte de ma page web sera forcément écrit en blanc ?

Non, pas obligatoirement. Si vous dites par la suite que vous voulez vos titres en rouge, ce style aura la priorité et vos titres seront donc en rouge. En revanche, si vous n'indiquez rien de particulier (comme on l'a fait tout à l'heure), alors vos titres hériteront de la couleur blanche. Cela ne fonctionne pas uniquement pour la couleur, entendons-nous bien. Toutes les propriétés CSS seront héritées : vous pouvez par exemple demander une mise en gras dans la balise `<body>` et tous vos titres et paragraphes seront en gras.

Exemple d'héritage avec la balise `<mark>`

On a tendance à croire qu'on ne peut modifier que la couleur de fond de la page. C'est faux : vous pouvez changer le fond de n'importe quel élément : vos titres, vos paragraphes, certains mots... Dans ce cas, ils apparaîtront surlignés (comme si on avait mis un coup de marqueur dessus).

Vous vous souvenez par exemple de la balise `<mark>` qui permet de mettre en valeur certains mots ? Utilisons-la à nouveau ici :

```
<h1>Qui a éteint la lumière ?</h1>

<p>Brr, il fait tout noir sur ce site, c'est un peu <mark>inquiétant</mark> comme ambiance non vous trouvez pas ?</p>
```

Par défaut, le texte s'affiche sur un fond jaune. Vous pouvez changer ce comportement en CSS :

```
body
{
    background-color: black;
    color: white;
}

mark
{
    /* La couleur de fond prend le pas sur celle de toute la page */
    background-color: red;
}
```

Sur le texte de la balise `<mark>`, c'est la couleur de fond rouge qui s'applique. En effet, même si le fond de la page est noir, c'est la propriété CSS de l'élément le plus précis qui a la priorité (figure 8.7).

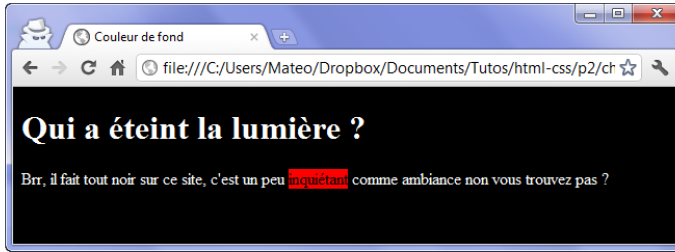


FIGURE 8.7 – Le CSS de la balise <mark> est prioritaire.

Le même principe vaut pour toutes les balises HTML et toutes les propriétés CSS ! Si vous dites :

- mes paragraphes ont une taille de 1.2 em ;
- mes textes importants (<strong>) ont une taille de 1.4 em ;

... on pourrait penser qu'il y a un conflit. Le texte important fait partie d'un paragraphe, quelle taille lui donner ? 1.2 em ou 1.4 em ? Le CSS décide que c'est la déclaration la plus précise qui l'emporte : comme <strong> correspond à un élément plus précis que les paragraphes, le texte sera écrit en 1.4 em.

Images de fond

Dans les exemples qui suivent, je vais modifier l'image de fond de la page. Cependant, tout comme pour la couleur de fond, n'oubliez pas que l'image de fond ne s'applique pas forcément à la page entière. On peut aussi mettre une image de fond derrière les titres, paragraphes, etc.

Appliquer une image de fond

La propriété permettant d'indiquer une image de fond est `background-image`. Comme valeur, on doit renseigner `url("nom_de_l_image.png")`. Par exemple :

```
body
{
    background-image: url("neige.png");
}
```

Ce qui nous donne la figure 8.8.

Bien entendu, votre fond n'est pas forcément en PNG, il peut aussi être en JPEG ou en GIF. L'adresse indiquant où se trouve l'image de fond peut être écrite en absolu (`http://...`) ou en relatif (`fond.png`).



Figure 8.8 – Une image de fond sur la page



Attention lorsque vous écrivez une adresse en relatif dans le fichier CSS ! L'adresse de l'image doit être indiquée *par rapport au fichier* `.css` et non pas par rapport au fichier `.html`. Pour simplifier les choses, je vous conseille de placer l'image de fond dans le même dossier que le fichier `.css` (ou dans un sous-dossier).

Options disponibles pour l'image de fond

On peut compléter la propriété `background-image` que nous venons de voir par plusieurs autres propriétés qui permettent de changer le comportement de l'image de fond.

`background-attachment` : fixer le fond

La propriété CSS `background-attachment` permet de « fixer » le fond. L'effet obtenu est intéressant car on voit alors le texte « glisser » par-dessus le fond. Deux valeurs sont disponibles :

- `fixed` : l'image de fond reste fixe ;
- `scroll` : l'image de fond défile avec le texte (par défaut).

```
body
{
    background-image: url("neige.png");
    background-attachment: fixed; /* Le fond restera fixe */
}
```

`background-repeat` : répétition du fond

Par défaut, l'image de fond est répétée en mosaïque. Vous pouvez changer cela avec la propriété `background-repeat` :

- `no-repeat` : le fond ne sera pas répété. L'image sera donc unique sur la page.
- `repeat-x` : le fond sera répété uniquement sur la première ligne, horizontalement.
- `repeat-y` : le fond sera répété uniquement sur la première colonne, verticalement.
- `repeat` : le fond sera répété en mosaïque (par défaut).

Exemple d'utilisation :

```
body
{
    background-image: url("soleil.png");
    background-repeat: no-repeat;
}
```

`background-position` : position du fond

On peut indiquer où doit se trouver l'image de fond avec `background-position`. Cette propriété n'est intéressante que si elle est combinée avec `background-repeat: no-repeat;` (un fond qui ne se répète pas).

Vous devez donner à `background-position` deux valeurs en pixels pour indiquer la position du fond par rapport au coin supérieur gauche de la page (ou du paragraphe, si vous appliquez le fond à un paragraphe). Ainsi, si vous tapez :

```
background-position: 30px 50px;
```

... votre fond sera placé à 30 pixels de la gauche et à 50 pixels du haut. Il est aussi possible d'utiliser ces valeurs en anglais :

- `top` : en haut ;
- `bottom` : en bas ;
- `left` : à gauche ;
- `center` : centré ;
- `right` : à droite.

Il est possible de combiner ces mots. Par exemple, pour aligner une image en haut à droite, vous taperez :

```
background-position: top right;
```

Ainsi, si je veux afficher un soleil en image de fond (figure 8.9), en un unique exemplaire (`no-repeat`), toujours visible (`fixed`) et positionné en haut à droite (`top right`), je vais écrire ceci :

```
body
{
    background-image: url("soleil.png");
    background-attachment: fixed; /* Le fond restera fixe */
```

```
    background-repeat: no-repeat; /* Le fond ne sera pas répété
        */
    background-position: top right; /* Le fond sera placé en
       haut à droite */
}
```



FIGURE 8.9 – Un soleil placé en image de fond, en haut à droite

Combiner les propriétés

Si vous utilisez beaucoup de propriétés en rapport avec le fond (comme c'est le cas sur ce dernier exemple), vous pouvez utiliser une sorte de « super-propriété » appelée `background` dont la valeur peut combiner plusieurs des propriétés vues précédemment : `background-image`, `background-repeat`, `background-attachment` et `background-position`.

On peut donc tout simplement écrire :

```
body
{
    background: url("soleil.png") fixed no-repeat top right;
}
```

C'est la première « super-propriété » que je vous montre, il y en aura d'autres. Il faut savoir que :

- L'ordre des valeurs n'a pas d'importance. Vous pouvez combiner les valeurs dans n'importe quel ordre.
- Vous n'êtes pas obligés de mettre toutes les valeurs. Ainsi, si vous ne voulez pas écrire `fixed`, vous pouvez l'enlever sans problème.

Plusieurs images de fond

Depuis CSS3, il est possible de donner plusieurs images de fond à un élément. Pour cela, il suffit de séparer les déclarations par une virgule, comme ceci :

```
body
{
    background: url("soleil.png") fixed no-repeat top right,
        url("neige.png") fixed;
}
```

La première image de cette liste sera placée par-dessus les autres (figure 8.10). Attention donc, l'ordre de déclaration des images a son importance : si vous inversez le soleil et la neige dans le code CSS précédent, vous ne verrez plus le soleil !



FIGURE 8.10 – Images de fond multiples

À noter que les images de fond multiples fonctionnent sur tous les navigateurs sauf sur les anciennes versions d'Internet Explorer, qui ne reconnaît cette fonctionnalité qu'à partir de la version 9 (IE9).



Une dernière chose avant d'en terminer avec les images de fond : dans tous ces exemples, j'ai appliqué un fond à la page entière (`body`). Mais cela ne doit pas vous faire oublier qu'on peut appliquer un fond à n'importe quel élément (un titre, un paragraphe, certains mots d'un paragraphe, etc.). Je vous conseille donc, pour vous entraîner, d'essayer d'appliquer un fond à vos titres ou paragraphes. Si vous avez un peu de goût (contrairement à moi !) vous arriverez certainement à donner une très belle allure à votre page web.

La transparence

Le CSS nous permet de jouer très facilement avec les niveaux de transparence des éléments ! Pour cela, nous allons utiliser des fonctionnalités de CSS3 : la propriété `opacity` et la notation RGBa.

La propriété opacity

La propriété `opacity`, très simple, permet d'indiquer le niveau d'opacité (c'est l'inverse de la transparence).

- Avec une valeur de 1, l'élément sera totalement opaque : c'est le comportement par défaut.
- Avec une valeur de 0, l'élément sera totalement transparent.

Il faut donc choisir une valeur comprise entre 0 et 1. Ainsi, avec une valeur de 0.6, votre élément sera opaque à 60%. . . et on verra donc à travers !

Voici comment on peut l'utiliser :

```
p
{
    opacity: 0.6;
}
```

Voici un exemple qui va nous permettre d'apprécier la transparence. Vous en trouverez le rendu à la figure 8.11.

```
body
{
    background: url('neige.png');
}

p
{
    background-color: black;
    color: white;
    opacity: 0.3;
}
```

Notez que la transparence fonctionne sur tous les navigateurs récents, y compris Internet Explorer à partir de IE9.



Si vous appliquez la propriété opacity à un élément de la page, *tout* le contenu de cet élément sera rendu transparent (même les images, les autres blocs à l'intérieur, etc.). Si vous voulez juste rendre la couleur de fond transparente, utilisez plutôt la notation RGBa que nous allons découvrir.



FIGURE 8.11 – Un paragraphe transparent avec `opacity`

La notation RGBa

CSS3 nous propose une autre façon de jouer avec la transparence : la notation RGBa. Il s'agit en fait de la notation RGB que nous avons vue précédemment, mais avec un quatrième paramètre : le niveau de transparence (appelé « canal alpha »). De la même façon que précédemment, avec une valeur de 1, le fond est complètement opaque. Avec une valeur inférieure à 1, il est transparent.

```
p
{
    background-color: rgba(255, 0, 0, 0.5); /* Fond rouge à
        moitié transparent */
}
```

C'est aussi simple que cela. Vous pouvez obtenir exactement le même effet qu'avec `opacity` juste en jouant avec la notation RGBa, essayez !

Cette notation est connue de tous les navigateurs récents, y compris Internet Explorer (à partir de IE9). Pour les navigateurs plus anciens, il est recommandé d'indiquer la notation RGB classique en plus de RGBa. Pour ces navigateurs, le fond ne sera alors pas transparent mais, au moins, il y aura bien une couleur d'arrière-plan.

```
p
{
    background-color: rgb(255,0,0); /* Pour les navigateurs
        anciens */
    background-color: rgba(255,0,0,0.5); /* Pour les
        navigateurs plus récents */
}
```

En résumé

- On change la couleur du texte avec la propriété `color`, la couleur de fond avec `background-color`.
- On peut indiquer une couleur en écrivant son nom en anglais (`black`, par exemple), sous forme hexadéciale (`#FFC8D3`) ou en notation RGB (`rgb(250,25,118)`).
- On peut ajouter une image de fond avec `background-image`. On peut choisir de fixer l'image de fond, de l'afficher en mosaïque ou non, et même de la positionner où on veut sur la page.
- On peut rendre une portion de la page transparente avec la propriété `opacity` ou avec la notation RGBa (identique à la notation RGB, avec une quatrième valeur indiquant le niveau de transparence).

Chapitre 9

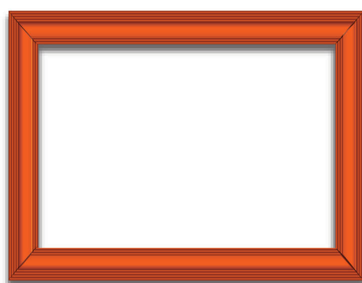
Les bordures et les ombres

Difficulté :

Nouveau chapitre, nouveau lot de propriétés CSS. Ici, nous allons nous intéresser aux bordures et aux effets d'ombrage que l'on peut appliquer, aussi bien sur le texte que sur les blocs qui constituent notre page.

Nous réutiliserons en particulier nos connaissances sur les couleurs pour choisir la couleur de nos bordures et de nos ombres.

Prêts à vous en mettre une nouvelle fois plein la vue ?



Bordures standard

Le CSS vous offre un large choix de bordures pour décorer votre page. De nombreuses propriétés CSS vous permettent de modifier l'apparence de vos bordures : `border-width`, `border-color`, `border-style`...

Pour aller à l'essentiel, je vous propose ici d'utiliser directement la super-propriété `border` qui regroupe l'ensemble de ces propriétés. Vous vous souvenez de la super-propriété `background` ? Cela fonctionne sur le même principe : on va pouvoir combiner plusieurs valeurs.

Pour `border` on peut utiliser jusqu'à trois valeurs pour modifier l'apparence de la bordure :

- **La largeur** : indiquez la largeur de votre bordure. Mettez une valeur en pixels (comme `2px`).
- **La couleur** : c'est la couleur de votre bordure. Utilisez, comme on l'a appris, soit un nom de couleur (`black`, `red`,...), soit une valeur hexadécimale (`#FF0000`), soit une valeur RGB (`rgb(198, 212, 37)`).
- **Le type de bordure** : là, vous avez le choix. Votre bordure peut être un simple trait, ou des pointillés, ou encore des tirets, etc. Voici les différentes valeurs disponibles :
  - `none` : pas de bordure (par défaut) ;
  - `solid` : un trait simple ;
  - `dotted` : pointillés ;
  - `dashed` : tirets ;
  - `double` : bordure double ;
  - `groove` : en relief ;
  - `ridge` : autre effet relief ;
  - `inset` : effet 3D global enfoncé ;
  - `outset` : effet 3D global surélevé.

Ainsi, pour avoir une bordure bleue, en tirets, épaisse de 3 pixels autour de mes titres, je vais écrire :

```
h1
{
    border: 3px blue dashed;
}
```

La figure 9.1 vous présente les différents styles de bordures que vous pouvez utiliser.

En haut, à droite, à gauche, en bas...

Qui a dit que vous étiez obligés d'appliquer la même bordure aux quatre côtés de votre élément ? Taratata, si vous voulez mettre des bordures différentes en fonction du côté (haut, bas, gauche ou droite), vous pouvez le faire sans problème. Dans ce cas, vous devrez utiliser ces quatre propriétés :

- `border-top` : bordure du haut ;
- `border-bottom` : bordure du bas ;

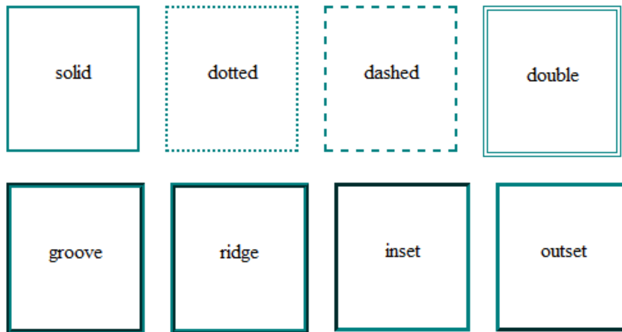


FIGURE 9.1 – Les différents types de bordures

- `border-left` : bordure de gauche ;
- `border-right` : bordure de droite.



Il existe aussi des équivalents pour paramétrer chaque détail de la bordure si vous le désirez : `border-top-width` pour modifier l'épaisseur de la bordure du haut, `border-top-color` pour la couleur du haut, etc.

Ce sont aussi des super-propriétés, elles fonctionnent comme `border` mais ne s'appliquent donc qu'à un seul côté.

Pour ajouter une bordure uniquement à gauche et à droite des paragraphes, on écrira donc :

```
p
{
    border-left: 2px solid black;
    border-right: 2px solid black;
}
```



On peut modifier les bordures de n'importe quel type d'élément sur la page. Nous l'avons fait ici sur les paragraphes mais on peut aussi modifier la bordure des images, des textes importants comme `<strong>`, etc.

Bordures arrondies

Les bordures arrondies, c'est un peu le Saint Graal attendu par les webmasters depuis des millénaires (ou presque). Depuis que CSS3 est arrivé, il est enfin possible d'en créer facilement !

La propriété `border-radius` va nous permettre d'arrondir facilement les angles de n'importe quel élément. Il suffit d'indiquer la taille (« l'importance ») de l'arrondi en

pixels :

```
p
{
    border-radius: 10px;
}
```

L'arrondi se voit notamment si l'élément a des bordures, comme sur la figure 9.2.

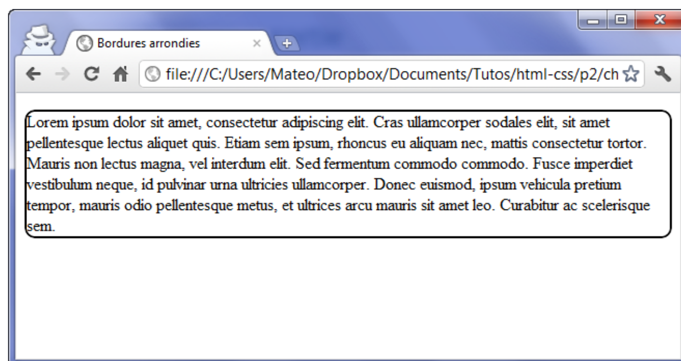


Figure 9.2 – Des bordures arrondies

. . . ou s'il a une couleur de fond, comme sur la figure 9.3.

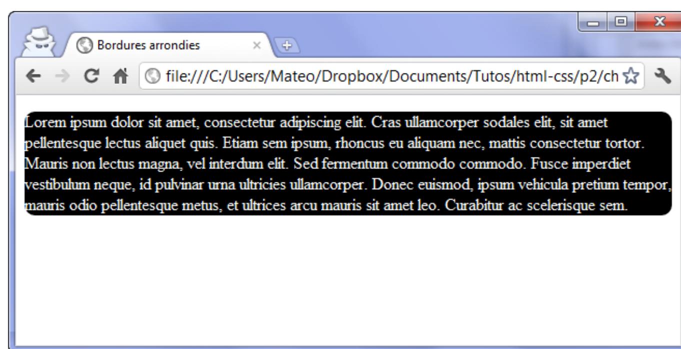


Figure 9.3 – Un fond aux coins arrondis

On peut aussi préciser la forme de l'arrondi pour chaque coin. Dans ce cas, indiquez quatre valeurs :

```
p
{
    border-radius: 10px 5px 10px 5px;
}
```

Les valeurs correspondent aux angles suivants dans cet ordre :

1. en haut à gauche ;
2. en haut à droite ;
3. en bas à droite ;
4. en bas à gauche.

Enfin, il est possible d'affiner l'arrondi de nos angles en créant des courbes elliptiques figure 9.4). Dans ce cas, il faut indiquer deux valeurs séparées par une barre oblique (*slash*, caractère /). Le mieux est certainement de tester pour voir l'effet :

```
p
{
    border-radius: 20px / 10px;
}
```

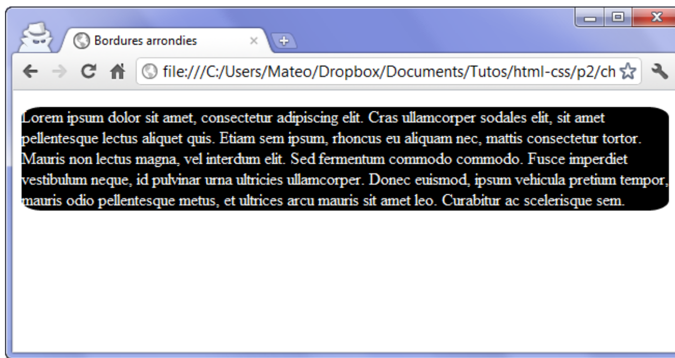


FIGURE 9.4 – Bordures arrondies elliptiques



Les bordures arrondies fonctionnent avec tous les navigateurs récents, y compris Internet Explorer à partir de la version 9 (IE9). Pour les anciennes versions de Mozilla Firefox, Chrome et Safari, il était nécessaire d'utiliser ce qu'on appelle des « préfixes vendeurs », c'est-à-dire qu'il fallait écrire dans le code CSS différentes versions de la propriété (`-moz-border-radius` pour Firefox, `-webkit-border-radius` pour Safari, etc.). Ce n'est heureusement plus nécessaire aujourd'hui, sauf si vous voulez gérer les anciennes versions de ces navigateurs.

Les ombres

Les ombres font partie des nouveautés récentes proposées par CSS3. Aujourd'hui, il suffit d'une seule ligne de CSS pour ajouter des ombres dans une page !

Nous allons ici découvrir deux types d'ombres :

– les ombres des boîtes ;
– les ombres du texte.

`box-shadow` : les ombres des boîtes

La propriété `box-shadow` s'applique à tout le bloc et prend quatre valeurs dans l'ordre suivant :

1. le décalage horizontal de l'ombre ;
2. le décalage vertical de l'ombre ;
3. l'adoucissement du dégradé ;
4. la couleur de l'ombre.

Par exemple, pour une ombre noire de 6 pixels, sans adoucissement, on écrira :

```
p
{
    box-shadow: 6px 6px 0px black;
}
```

Cela donne le résultat illustré à la figure 9.5 (j'ai ajouté une bordure au paragraphe pour qu'on voie mieux l'effet).

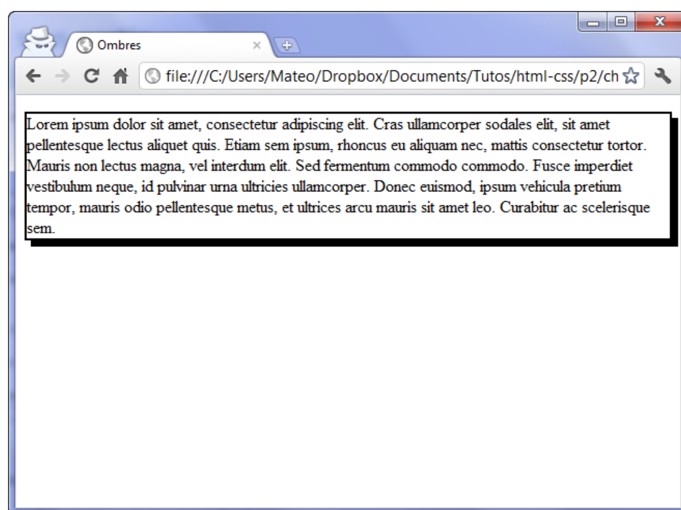


FIGURE 9.5 – Une ombre sous le paragraphe

Ajoutons un adoucissement grâce au troisième paramètre (figure 9.6). L'adoucissement peut être faible (inférieur au décalage), normal (égal au décalage) ou élevé (supérieur au décalage). Essayons un décalage normal :

```
p
{
    box-shadow: 6px 6px 6px black;
}
```

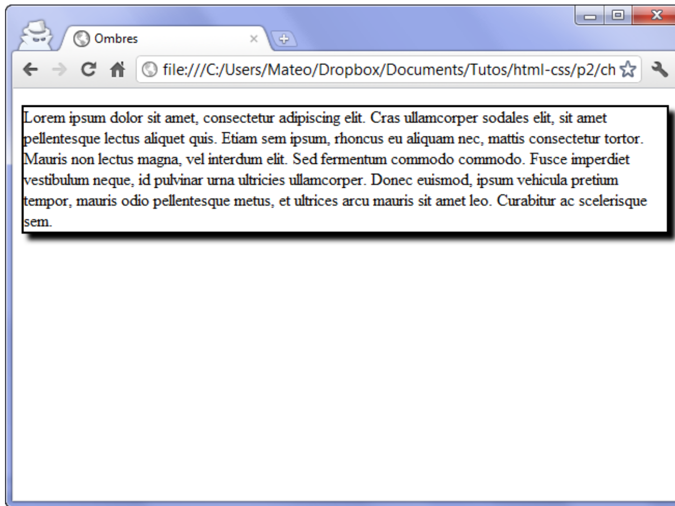


FIGURE 9.6 – Une ombre adoucie sous le paragraphe

On peut aussi rajouter une quatrième valeur facultative : `inset`. Dans ce cas, l'ombre sera placée à l'intérieur du bloc, pour donner un effet enfoncé :

```
p
{
    box-shadow: 6px 6px 6px black inset;
}
```

Je vous laisse essayer de voir le résultat.



La propriété `box-shadow` fonctionne sur tous les navigateurs récents, IE9 inclus. Pour certains navigateurs, en particulier les navigateurs mobiles, il faut encore rajouter un préfixe. Ainsi, il faudra écrire une version `-webkit-box-shadow` pour que cela fonctionne sur les navigateurs Android et iOS.

`text-shadow` : l'ombre du texte

Avec `text-shadow`, vous pouvez ajouter une ombre directement sur les lettres de votre texte ! Les valeurs fonctionnent exactement de la même façon que `box-shadow` : décalage, adoucissement et couleur.

```
p
{
    text-shadow: 2px 2px 4px black;
}
```

Le résultat est illustré à la figure 9.7.

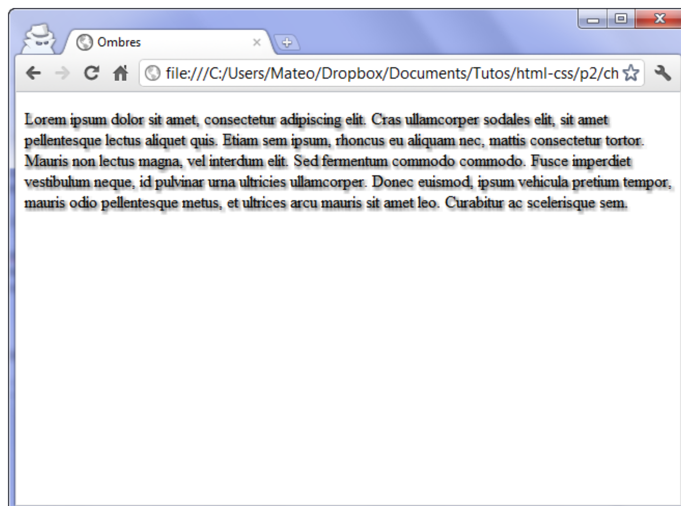


Figure 9.7 – Texte ombré



Cette propriété est reconnue par tous les navigateurs récents, sauf Internet Explorer, qui ne la reconnaît qu'à partir de la version IE10.

En résumé

- On peut appliquer une bordure à un élément avec la propriété `border`. Il faut indiquer la largeur de la bordure, sa couleur et son type (trait continu, pointillés...).
- On peut arrondir les bordures avec `border-radius`.
- On peut ajouter une ombre aux blocs de texte avec `box-shadow`. On doit indiquer le décalage vertical et horizontal de l'ombre, son niveau d'adoucissement et sa couleur.
- Le texte peut lui aussi avoir une ombre avec `text-shadow`.

Chapitre 10

Création d'apparences dynamiques

Difficulté :

C'est une de ses forces : le CSS nous permet aussi de modifier l'apparence des éléments de façon dynamique, c'est-à-dire que des éléments peuvent changer de forme une fois que la page a été chargée. Nous allons faire appel à une fonctionnalité puissante du CSS : les pseudo-formats.

Nous verrons dans ce chapitre comment changer l'apparence :

- au survol ;
- lors du clic ;
- lors du focus (élément sélectionné) ;
- lorsqu'un lien a été consulté.

Vous allez voir que le langage CSS n'a pas fini de nous étonner !



Au survol

Nous allons découvrir dans ce chapitre plusieurs pseudo-formats CSS. Le premier que je vais vous montrer s'appelle `:hover`. Comme tous les autres pseudo-formats que nous allons voir, c'est une information que l'on rajoute après le nom de la balise (ou de la classe) dans le CSS, comme ceci :

```
a:hover
{

}
```

`:hover` signifie « survoler ». `a:hover` peut donc se traduire par : « Quand la souris est sur le lien » (quand on pointe dessus).

À partir de là, c'est à vous de définir l'apparence que doivent avoir les liens lorsqu'on pointe dessus. Laissez libre cours à votre imagination, il n'y a pas de limite.

Voici un exemple de présentation des liens, mais n'hésitez pas à inventer le vôtre :

```
a /* Liens par défaut (non survolés) */
{
    text-decoration: none;
    color: red;
    font-style: italic;
}

a:hover /* Apparence au survol des liens */
{
    text-decoration: underline;
    color: green;
}
```

On a défini ici deux versions des styles pour les liens :

– pour les liens par défaut (non survolés) ;
– pour les liens au survol.

Le résultat se trouve à la figure 10.1.

Sympa, n'est-ce pas ?

Même si on l'utilise souvent sur les liens, vous pouvez modifier l'apparence de n'importe quel élément. Par exemple, vous pouvez modifier l'apparence des paragraphes lorsqu'on pointe dessus :

```
p:hover /* Quand on pointe sur un paragraphe */
{

}
```



FIGURE 10.1 – Changement d'apparence au survol de la souris

Au clic et lors de la sélection

Vous pouvez interagir encore plus finement en CSS. Nous allons voir ici que nous pouvons changer l'apparence des éléments lorsque l'on clique dessus et lorsqu'ils sont sélectionnés !

`:active` : au moment du clic

Le pseudo-format `:active` permet d'appliquer un style particulier *au moment du clic*. En pratique, il n'est utilisé que sur les liens.

Le lien gardera cette apparence très peu de temps : en fait, le changement intervient lorsque le bouton de la souris est enfoncé. En clair, ce n'est pas forcément toujours bien visible.

On peut par exemple changer la couleur de fond du lien lorsque l'on clique dessus :

```
a:active /* Quand le visiteur clique sur le lien */
{
    background-color: #FFCC66;
}
```

`:focus` : lorsque l'élément est sélectionné

Là, c'est un peu différent. Le pseudo-format `:focus` applique un style *lorsque l'élément est sélectionné*.



C'est-à-dire ?

Une fois que vous avez cliqué, le lien reste « sélectionné » (il y a une petite bordure en pointillés autour). C'est cela, la sélection.



Ce pseudo-format pourra être appliqué à d'autres balises HTML que nous n'avons pas encore vues, comme les éléments de formulaires.

Essayons pour l'instant sur les liens :

```
a:focus /* Quand le visiteur sélectionne le lien */
{
    background-color: #FFCC66;
}
```



Sous Google Chrome et Safari, l'effet ne se voit que si l'on appuie sur la touche Tab.

Lorsque le lien a déjà été consulté

Il est possible d'appliquer un style à un lien vers une page qui a déjà été vue. Par défaut, le navigateur colore le lien en un violet assez laid (de mon point de vue du moins !).

Vous pouvez changer cette apparence avec `:visited` (qui signifie « visité »). En pratique, sur les liens consultés, on ne peut pas changer beaucoup de choses à part la couleur (figure 10.2).

```
a:visited /* Quand le visiteur a déjà vu la page concernée */
{
    color: #AAA; /* Appliquer une couleur grise */
}
```

Si vous ne souhaitez pas que les liens déjà visités soient colorés d'une façon différente, il vous faudra leur appliquer la même couleur qu'aux liens normaux. De nombreux sites web font cela (le Site du Zéro y compris !). Une exception notable : Google... ce qui



FIGURE 10.2 – Liens visités en gris

est plutôt pratique, puisque l'on peut voir dans les résultats d'une recherche si on a déjà consulté ou non les sites que Google nous présente.

En résumé

- En CSS, on peut modifier l'apparence de certaines sections dynamiquement, après le chargement de la page, lorsque certains évènements se produisent. On utilise pour cela les pseudo-formats.
- Le pseudo-format `:hover` permet de changer l'apparence au survol (par exemple : `a:hover` pour modifier l'apparence des liens lorsque la souris pointe dessus).
- Le pseudo-format `:active` modifie l'apparence des liens au moment du clic, `:visited` lorsqu'un lien a déjà été visité.
- Le pseudo-format `:focus` permet de modifier l'apparence d'un élément sélectionné.

Troisième partie

Mise en page du site

Chapitre 11

Structurer sa page

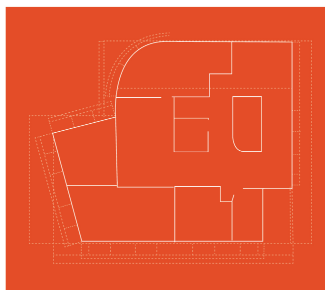
Difficulté :

Nous approchons de plus en plus du but. Si nos pages web ne ressemblent pas encore tout à fait aux sites web que nous connaissons, c'est qu'il nous manque les connaissances nécessaires pour faire la mise en page.

En général, une page web est constituée d'un en-tête (tout en haut), de menus de navigation (en haut ou sur les côtés), de différentes sections au centre... et d'un pied de page (tout en bas).

Dans ce chapitre, nous allons nous intéresser aux nouvelles balises HTML dédiées à la structuration du site. Ces balises ont été introduites par HTML5 (elles n'existaient pas avant) et vont nous permettre de dire : « Ceci est mon en-tête », « Ceci est mon menu de navigation », etc.

Pour le moment, nous n'allons pas encore faire de mise en page. Nous allons en fait préparer notre document HTML pour pouvoir découvrir la mise en page dans les prochains chapitres.



Les balises structurantes de HTML5

Je vais vous présenter ici les nouvelles balises introduites par HTML5 pour structurer nos pages. Vous allez voir, cela ne va pas beaucoup changer l'apparence de notre site pour le moment, mais il sera bien construit et prêt à être mis en forme ensuite !

<header> : l'en-tête

La plupart des sites web possèdent en général un en-tête, appelé *header* en anglais. On y trouve le plus souvent un logo, une bannière, le slogan de votre site...

Vous devrez placer ces informations à l'intérieur de la balise <header> :

```
<header>
    <!-- Placez ici le contenu de l'en-tête de votre page -->
</header>
```

La figure 11.1, par exemple, représente une partie du Site du Zéro et vous pouvez voir son en-tête.

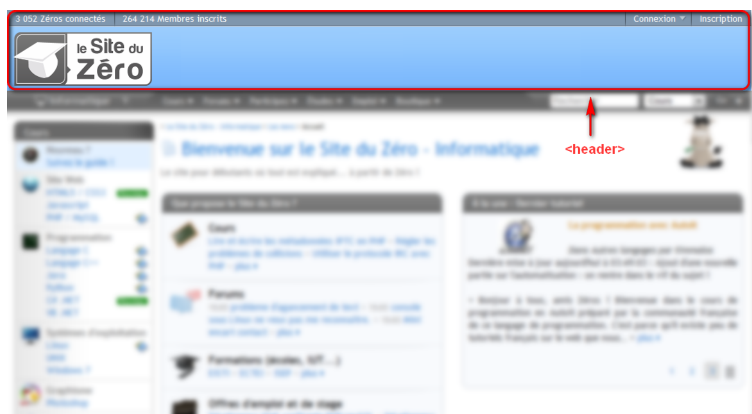


FIGURE 11.1 – L'en-tête du Site du Zéro

L'en-tête peut contenir tout ce que vous voulez : images, liens, textes...



Il peut y avoir plusieurs en-têtes dans votre page. Si celle-ci est découpée en plusieurs sections, chaque section peut en effet avoir son propre <header>.

<footer> : le pied de page

À l'inverse de l'en-tête, le pied de page se trouve en général tout en bas du document. On y trouve des informations comme des liens de contact, le nom de l'auteur, les

mentions légales, etc.

```
<footer>
    <!-- Placez ici le contenu du pied de page -->
</footer>
```

La figure 11.2 vous montre à quoi ressemble le pied de page du Site du Zéro.

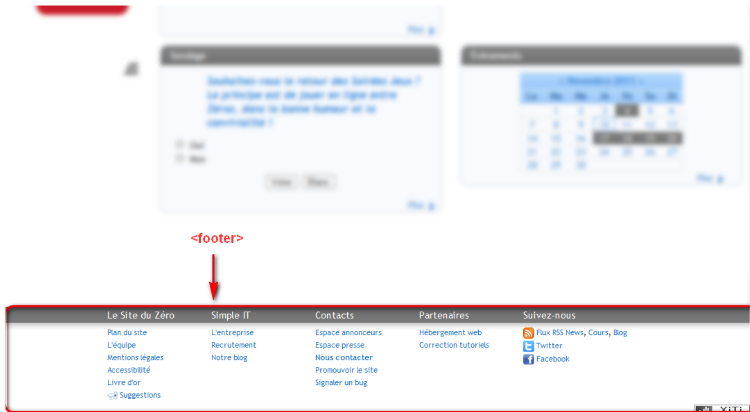


FIGURE 11.2 – Pied de page du Site du Zéro

<nav> : principaux liens de navigation

La balise `<nav>` doit regrouper tous les principaux liens de navigation du site. Vous y placerez par exemple le menu principal de votre site.

Généralement, le menu est réalisé sous forme de liste à puces à l'intérieur de la balise `<nav>` :

```
<nav>
    <ul>
        <li><a href="index.html">Accueil</a></li>
        <li><a href="forum.html">Forum</a></li>
        <li><a href="contact.html">Contact</a></li>
    </ul>
</nav>
```

En figure 11.3, vous pouvez observer le menu principal de navigation du Site du Zéro, qui pourrait utiliser la balise `<nav>`.

<section> : une section de page

La balise `<section>` sert à regrouper des contenus en fonction de leur thématique. Elle englobe généralement une portion du contenu au centre de la page.

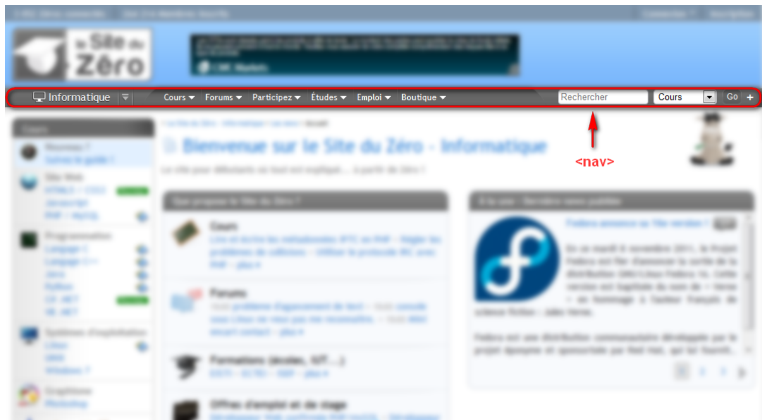


FIGURE 11.3 – Le menu de navigation du Site du Zéro

```
<section>
    <h1>Ma section de page</h1>
    <p>Bla bla bla</p>
</section>
```

Sur la page d'accueil du portail Free.fr, on trouve plusieurs blocs qui pourraient être considérés comme des sections de page (figure 11.4).

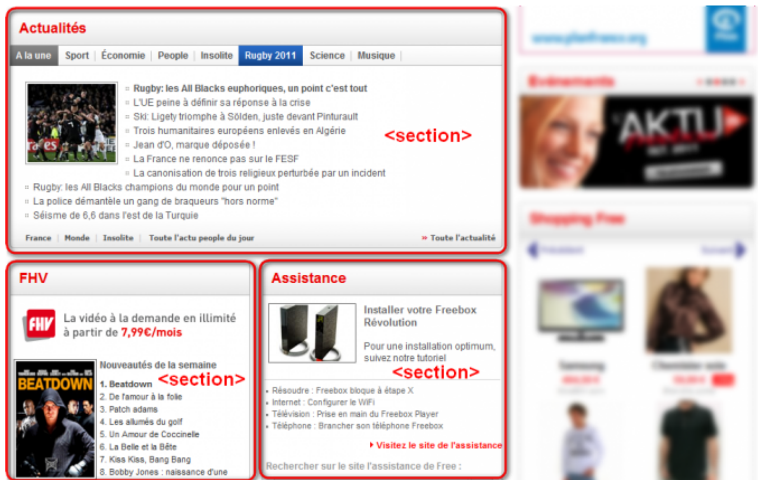


FIGURE 11.4 – Des sections de page sur le portail de Free



Chaque section peut avoir son titre de niveau 1 (<h1>), de même que l'en-tête peut contenir un titre <h1> lui aussi. Chacun de ces blocs étant indépendant des autres, il n'est pas illogique de retrouver plusieurs titres <h1> dans le code de la page web. On a ainsi « Le titre <h1> du <header> », « Le titre <h1> de cette <section> », etc.

<aside> : informations complémentaires

La balise <aside> est conçue pour contenir des informations complémentaires au document que l'on visualise. Ces informations sont généralement placées sur le côté (bien que ce ne soit pas une obligation).

```html
<aside>
    <!-- Placez ici des informations complémentaires -->
</aside>
```

Il peut y avoir plusieurs blocs <aside> dans la page.

Sur Wikipédia, par exemple, il est courant de voir à droite un bloc d'informations complémentaires à l'article que l'on visualise. Ainsi, sur la page présentant la planète Saturne (figure 11.5), on trouve dans ce bloc les caractéristiques de la planète (dimensions, masse, etc.).

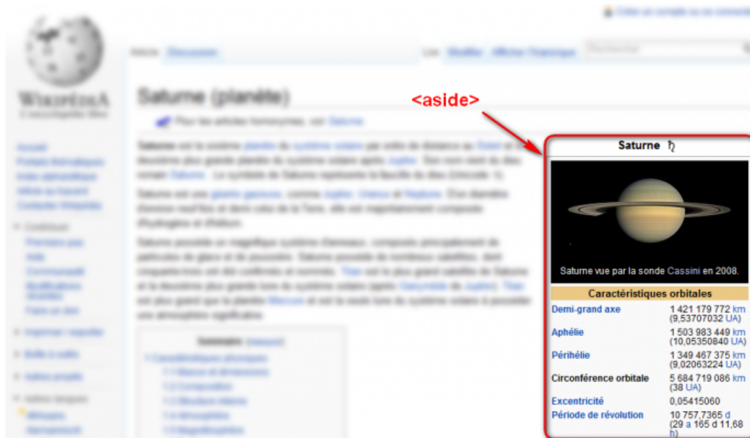


FIGURE 11.5 – Bloc d'informations complémentaires sur Wikipédia

<article> : un article indépendant

La balise <article> sert à englober une portion généralement autonome de la page. C'est une partie de la page qui pourrait ainsi être reprise sur un autre site. C'est le cas par exemple des actualités (articles de journaux ou de blogs).

```
<article>
    <h1>Mon article</h1>
    <p>Bla bla bla bla</p>
</article>
```

Par exemple, voici un article sur le Site du Zéro en figure 11.6.



FIGURE 11.6 – Un article publié sur le Site du Zéro

.

Résumé

Ouf, cela fait beaucoup de nouvelles balises à retenir. Heureusement, je vous ai fait un petit schéma (figure 11.7) pour vous aider à retenir leur rôle !



Ne vous y trompez pas : ce schéma propose un *exemple* d'organisation de la page. Rien ne vous empêche de décider que votre menu de navigation est à droite, ou tout en haut, que vos balises `<aside>` sont au-dessus, etc. On peut même imaginer une seconde balise `<header>`, placée cette fois à l'intérieur d'une `<section>`. Dans ce cas-là, elle sera considérée comme étant l'en-tête de la section. Enfin, une section ne doit pas forcément contenir un `<article>` et des `<aside>`. Utilisez ces balises uniquement si vous en avez besoin. Rien ne vous interdit de créer des sections contenant seulement des paragraphes, par exemple.

Exemple concret d'utilisation des balises

Essayons d'utiliser les balises que nous venons de découvrir pour structurer notre page web. Le code ci-dessous reprend toutes les balises que nous venons de voir au sein d'une

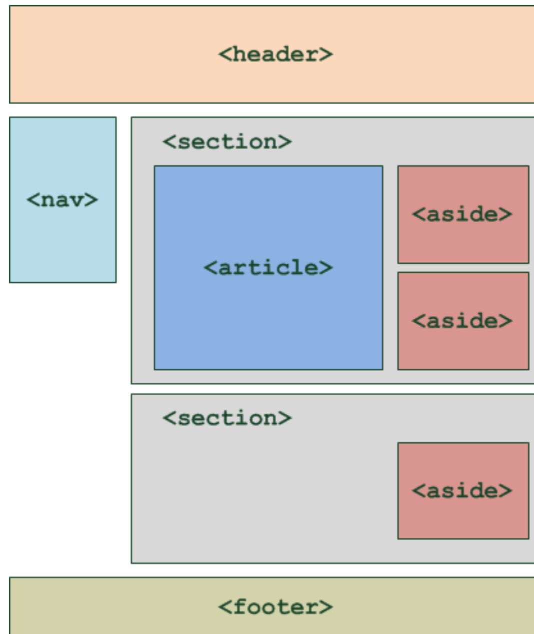


FIGURE 11.7 – Sections de la page identifiées par les balises

page web complète :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Zozor - Le Site Web</title>
    </head>

    <body>
        <header>
            <h1>Zozor</h1>
            <h2>Carnets de voyage</h2>
        </header>

        <nav>
            <ul>
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Blog</a></li>
                <li><a href="#">CV</a></li>
            </ul>
        </nav>

        <section>
            <aside>
```

```
                    <h1>À propos de l'auteur</h1>
                    <p>C'est moi, Zozor ! Je suis né un 23 novembre
                        2005.</p>
                </aside>
                <article>
                    <h1>Je suis un grand voyageur</h1>
                    <p>Bla bla bla (texte de l'article)</p>
                </article>
            </section>

            <footer>
                <p>Copyright Zozor - Tous droits réservés<br />
                <a href="#">Me contacter !</a></p>
            </footer>

        </body>
</html>
```

 Copier ce code
Code web : 829837

Ce code peut vous aider à comprendre comment les balises doivent être agencées. Vous y reconnaissez un en-tête, un menu de navigation, un pied de page... et, au centre, une section avec un article et un bloc <aside> donnant des informations sur l'auteur de l'article.



À quoi ressemble la page que nous venons de créer ?

À rien ! Si vous testez le résultat, vous verrez juste du texte noir sur fond blanc (figure 11.8). C'est normal, il n'y a pas de CSS ! Par contre, la page est bien structurée, ce qui va nous être utile pour la suite.



Les liens sont volontairement factices (d'où la présence d'un simple #), ils n'amènent donc nulle part (eh, c'est juste une page de démo) !



Je ne comprends pas l'intérêt de ces balises. On peut très bien obtenir le même résultat sans les utiliser !

C'est vrai. En fait, ces balises sont seulement là pour expliquer à l'ordinateur « Ceci est l'en-tête », « Ceci est mon pied de page », etc. Elles n'indiquent pas, contrairement à ce qu'on pourrait penser, où doit être placé le contenu. C'est le rôle du CSS, comme nous le verrons dans peu de temps maintenant.



FIGURE 11.8 – Une page bien structurée mais sans CSS

À l'heure actuelle, pour tout vous dire, ces balises ont encore assez peu d'utilité. On pourrait très bien utiliser des balises génériques `<div>` à la place pour englober les différentes portions de notre contenu. D'ailleurs, c'est comme cela qu'on faisait avant l'arrivée de ces nouvelles balises HTML5.

Néanmoins, il est assez probable que, dans un futur proche, les ordinateurs commencent à tirer parti intelligemment de ces nouvelles balises. On peut imaginer par exemple un navigateur qui choisisse d'afficher les liens de navigation `<nav>` de manière toujours visible ! Quand l'ordinateur « comprend » la structure de la page, tout devient possible.

Assurer la compatibilité avec IE

Les nouvelles balises que nous venons de voir ne sont reconnues par Internet Explorer que depuis sa version 9 (IE9). Cela va poser un problème car, quand les anciennes versions d'IE ne connaissent pas une balise... elles ne traitent pas correctement la page (impossible de modifier leur CSS par exemple) !

Cela peut heureusement se régler assez facilement à l'aide d'un script JavaScript. Les scripts sont des petits morceaux de code qui permettent de manipuler la page web et d'effectuer certaines actions. Nous n'allons pas nous intéresser ici au JavaScript (ce n'est pas le sujet de ce cours) mais il faut savoir que ce type de script est appelé depuis les pages HTML d'une façon assez similaire à l'appel des fichiers CSS.

Les fichiers JavaScript portent généralement l'extension `.js`. Dans le code HTML, on les place en général dans la balise `<head>` avec cette balise :

```
<script src="monscript.js"></script>
```

Si je vous présente (brièvement) JavaScript ici, c'est parce qu'un petit script répondant au doux nom de `HTML5shiv` permet de faire en sorte que les balises que nous venons de voir (`<header>`, `<footer>`, `<section>`...) s'affichent correctement sur les anciennes versions d'Internet Explorer (IE6, IE7, IE8). Concrètement, il vous suffit d'ajouter les lignes suivantes dans votre code :

```
<!--[if lt IE 9]>
    <script src="http://html5shiv.googlecode.com/svn/trunk/
        html5.js"></script>
<![endif]-->
```

▷ Copier ce code
Code web : 317933

Placez ce code dans la balise `<head>` comme ceci :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <!--[if lt IE 9]>
            <script src="http://html5shiv.googlecode.com/svn/
                trunk/html5.js"></script>
        <![endif]-->
        <title>Inclusion de HTML5shiv</title>
    </head>

    <body>

    </body>
</html>
```

Et voilà, votre site s'affichera désormais sans problème dans Internet Explorer !



Sans rentrer dans le détail, sachez que `<!--[if lt IE 9]>` est un **commentaire conditionnel**. C'est un commentaire spécial qui n'est lu que par Internet Explorer. Il permet de faire en sorte que le script s'exécute uniquement sur les versions d'Internet Explorer inférieures à IE9 (les autres navigateurs n'en ont pas besoin et ignoreront le commentaire).

En résumé

- Plusieurs balises ont été introduites avec HTML5 pour délimiter les différentes zones qui constituent la page web :

  - `<header>` : en-tête ;
  - `<footer>` : pied de page ;
  - `<nav>` : liens principaux de navigation ;
  - `<section>` : section de page ;
  - `<aside>` : informations complémentaires ;
  - `<article>` : article indépendant.
- Ces balises peuvent être imbriquées les unes dans les autres. Ainsi, une section peut avoir son propre en-tête.
- Ces balises ne s'occupent pas de la mise en page. Elles servent seulement à indiquer à l'ordinateur le sens du texte qu'elles contiennent. On pourrait très bien placer l'en-tête en bas de la page si on le souhaite.
- Le code JavaScript `HTML5shiv` permet de faire en sorte que ces balises soient reconnues pour les versions d'Internet Explorer antérieures à IE9.

Chapitre 12

Le modèle des boîtes

Difficulté :

Une page web peut être vue comme une succession et un empilement de boîtes, qu'on appelle « blocs ». La plupart des éléments vus au chapitre précédent sont des blocs : `<header>`, `<article>`, `<nav>`... Mais nous connaissions déjà d'autres blocs : les paragraphes `<p>`, les titres `<h1>`...

Dans ce chapitre, nous allons apprendre à manipuler ces blocs comme de véritables boîtes. Nous allons leur donner des dimensions, les agencer en jouant sur leurs marges, mais aussi apprendre à gérer leur contenu... pour éviter que le texte ne dépasse de ces blocs !

Ce sont des notions fondamentales dont nous allons avoir besoin pour mettre en page notre site web... Soyez attentifs !



Les balises de type block et inline

En HTML, la plupart des balises peuvent se ranger dans l'une ou l'autre de deux catégories :

– Les balises **inline** : c'est le cas par exemple des liens `<a></a>`.
– Les balises **block** : c'est le cas par exemple des paragraphes `<p></p>`.



Il existe en fait plusieurs autres catégories très spécifiques, par exemple pour les cellules de tableau (type `table-cell`) ou les puces (type `list-item`). Nous n'allons pas nous y intéresser pour le moment car ces balises sont minoritaires.



Mais comment je reconnais une balise inline d'une balise block ?

C'est en fait assez facile :

– **block** : une balise de type block sur votre page web crée automatiquement un retour à la ligne avant et après. Il suffit d'imaginer tout simplement un bloc. Votre page web sera en fait constituée d'une série de blocs les uns à la suite des autres. Mais vous verrez qu'en plus, il est possible de mettre un bloc à l'intérieur d'un autre, ce qui va augmenter considérablement nos possibilités pour créer le design de notre site !
– **inline** : une balise de type inline se trouve obligatoirement à l'intérieur d'une balise block. Une balise inline ne crée pas de retour à la ligne, le texte qui se trouve à l'intérieur s'écrit donc à la suite du texte précédent, sur la même ligne (c'est pour cela que l'on parle de balise « en ligne »).

Pour bien visualiser le concept, voici en figure 12.1 un petit schéma que je vous ai concocté.

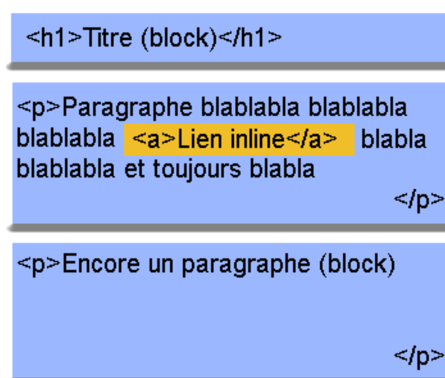


FIGURE 12.1 – Différence entre une balise `inline` et une balise `block`

– Sur fond bleu, vous avez tout ce qui est de type block.
– Sur fond jaune, vous avez tout ce qui est de type inline.

Comme vous pouvez le voir, les blocs sont les uns en-dessous des autres. On peut aussi les imbriquer les uns à l'intérieur des autres (souvenez-vous, nos blocs `<section>` contiennent par exemple des blocs `<aside>` !). La balise inline `<a></a>`, elle, se trouve à l'intérieur d'une balise block et le texte vient s'insérer sur la même ligne.

Quelques exemples

Afin de mieux vous aider à assimiler quelles balises sont inline et quelles balises sont block, voici un petit tableau dressant la liste de quelques balises courantes.

Balises block	Balises inline
<p>	<em>
<footer>	<strong>
<h1>	<mark>
<h2>	<a>
<article>	<img />
...	...

Ce tableau n'est pas complet, loin de là. Si vous voulez avoir la liste complète des balises qui existent et savoir si elles sont de type inline ou block, reportez-vous à l'annexe donnant la liste des balises HTML (page 285).

Les balises universelles

Vous les connaissez déjà car je vous les ai présentées il y a quelques chapitres. Ce sont des balises qui n'ont aucun sens particulier (contrairement à `<p>` qui veut dire « paragraphe », `<strong>` « important », etc.). Le principal intérêt de ces balises est que l'on peut leur appliquer une `class` (ou un `id`) pour le CSS quand aucune autre balise ne convient.

Il existe deux balises génériques et, comme par hasard, la seule différence entre les deux est que l'une d'elle est inline et l'autre est block :

– `<span></span>` (**inline**) ;
– `<div></div>` (**block**).

Respectez la sémantique !

Les balises universelles sont « pratiques » dans certains cas, certes, mais attention à ne pas en abuser. Je tiens à vous avertir de suite : beaucoup de webmasters mettent des `<div>` et des `<span>` trop souvent et oublient que d'autres balises plus adaptées existent.

Voici deux exemples :

- **Exemple d'un span inutile** : `<span class="important">`. Je ne devrais jamais voir ceci dans un de vos codes alors qu'il existe la balise `<strong>` qui sert à indiquer l'importance !
- **Exemple d'un div inutile** : `<div class="titre">`. Ceci est complètement absurde puisqu'il existe des balises faites spécialement pour les titres (`<h1>`, `<h2>`. . .).

Oui, vous allez me dire qu'au final le résultat (visuel) est le même. Je suis tout à fait d'accord. Mais les balises génériques n'apportent aucun sens à la page et ne peuvent pas être comprises par l'ordinateur. Utilisez toujours d'autres balises plus adaptées quand c'est possible. Google lui-même le conseille pour vous aider à améliorer la position de vos pages au sein de ses résultats de recherche !

Les dimensions

Nous allons ici travailler uniquement sur des balises de type block.

Pour commencer, intéressons-nous à la taille des blocs. Contrairement à un inline, un bloc a des dimensions précises. Il possède une largeur et une hauteur. Ce qui fait, ô surprise, qu'on dispose de deux propriétés CSS :

- `width` : c'est la largeur du bloc. À exprimer en pixels (px) ou en pourcentage (%).
- `height` : c'est la hauteur du bloc. Là encore, on l'exprime soit en pixels (px), soit en pourcentage (%).

Par défaut, un bloc prend 100% de la largeur disponible. On peut le vérifier en appliquant à nos blocs des bordures ou une couleur de fond (figure 12.2).

Maintenant, rajoutons un peu de CSS afin de modifier la largeur des paragraphes. Le CSS suivant dit : « Je veux que tous mes paragraphes aient une largeur de 50% ».

```
p
{
    width: 50%;
}
```

Le résultat est visible à la figure 12.3.

Les pourcentages seront utiles pour créer un design qui s'adapte automatiquement à la résolution d'écran du visiteur. Toutefois, il se peut que vous ayez besoin de créer des blocs ayant une dimension précise en pixels :

```
p
{
    width: 250px;
}
```

Minimum et maximum

On peut demander à ce qu'un bloc ait des dimensions minimales et maximales. C'est très pratique car cela nous permet de définir des dimensions « limites » pour que notre

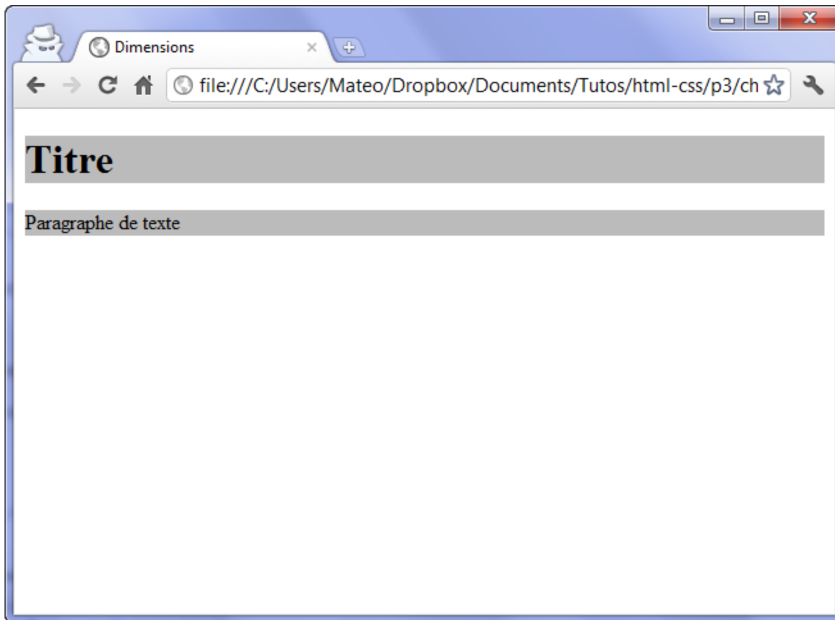


FIGURE 12.2 – Les blocs prennent toute la largeur disponible

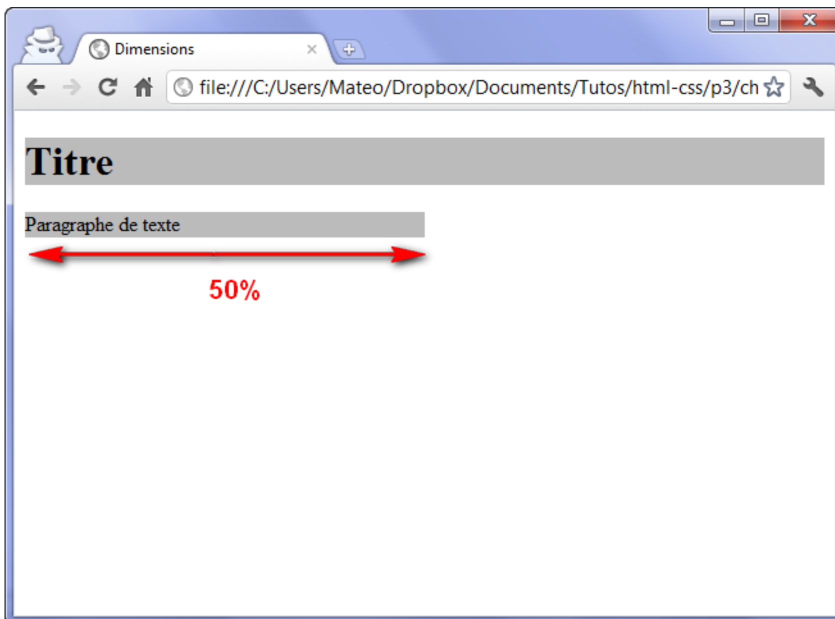


FIGURE 12.3 – Un paragraphe de 50% de largeur

site s'adapte aux différentes résolutions d'écran de nos visiteurs :

- `min-width` : largeur minimale ;
- `min-height` : hauteur minimale ;
- `max-width` : largeur maximale ;
- `max-height` : hauteur maximale.

Par exemple, on peut demander à ce que les paragraphes occupent 50% de la largeur *et* exiger qu'il fassent au moins 400 pixels de large dans tous les cas :

```
p
{
    width: 50%;
    min-width: 400px;
}
```

Observez le résultat en modifiant la largeur de la fenêtre de votre navigateur. Vous allez voir que, si celle-ci est trop petite, le paragraphe se force à occuper au moins 400 pixels de largeur.

Les marges

Il faut savoir que tous les blocs possèdent des marges. Il existe *deux types de marges* :

- les marges intérieures ;
- les marges extérieures.

Regardez bien le schéma qui se trouve à la figure 12.4.

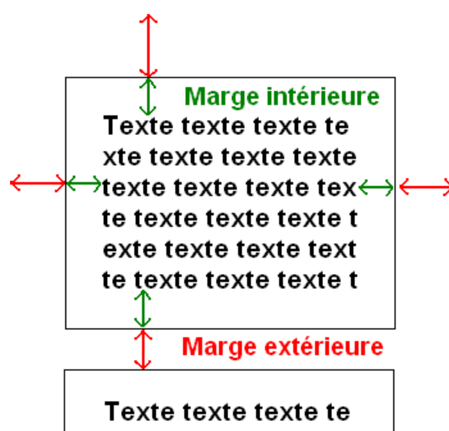


FIGURE 12.4 – Marges extérieure et intérieure

Sur ce bloc, j'ai mis une bordure pour qu'on repère mieux ses frontières.

- L'espace entre le texte et la bordure est la marge intérieure (en vert).
- L'espace entre la bordure et le bloc suivant est la marge extérieure (en rouge).

En CSS, on peut modifier la taille des marges avec les deux propriétés suivantes :

- `padding` : indique la taille de la marge intérieure. À exprimer en général en pixels (px).
- `margin` : indique la taille de la marge extérieure. Là encore, on utilise le plus souvent des pixels.



Les balises de type inline possèdent également des marges. Vous pouvez donc aussi essayer ces manipulations sur ce type de balises.

Pour bien voir les marges, prenons deux paragraphes auxquels j'applique simplement une petite bordure (figure 12.5) :

```
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
}
```



FIGURE 12.5 – Marges par défaut sur les paragraphes

Comme vous pouvez le constater, il n'y a par défaut pas de marge intérieure (`padding`). En revanche, il y a une marge extérieure (`margin`). C'est cette marge qui fait que deux paragraphes ne sont pas collés et qu'on a l'impression de « sauter une ligne ».



Les marges par défaut ne sont pas les mêmes pour toutes les balises de type block. Essayez d'appliquer ce CSS à des balises <div> qui contiennent du texte, par exemple : vous verrez que, dans ce cas, il n'y a par défaut ni marge intérieure, ni marge extérieure !

Supposons que je veuille rajouter une marge intérieure de 12 px aux paragraphes (figure 12.6) :

```
p
{
    width: 350px;
    border: 1px solid black;
    text-align: justify;
    padding: 12px; /* Marge intérieure de 12px */
}
```



FIGURE 12.6 – Une marge intérieure ajoutée aux paragraphes

Maintenant, je veux que mes paragraphes soient plus espacés entre eux. Je rajoute la propriété `margin` pour demander à ce qu'il y ait 50 px de marge entre deux paragraphes (figure 12.7) :

```
p
{
   width: 350px;
   border: 1px solid black;
```

```
    text-align: justify;
    padding: 12px;
    margin: 50px; /* Marge extérieure de 50px */
}
```

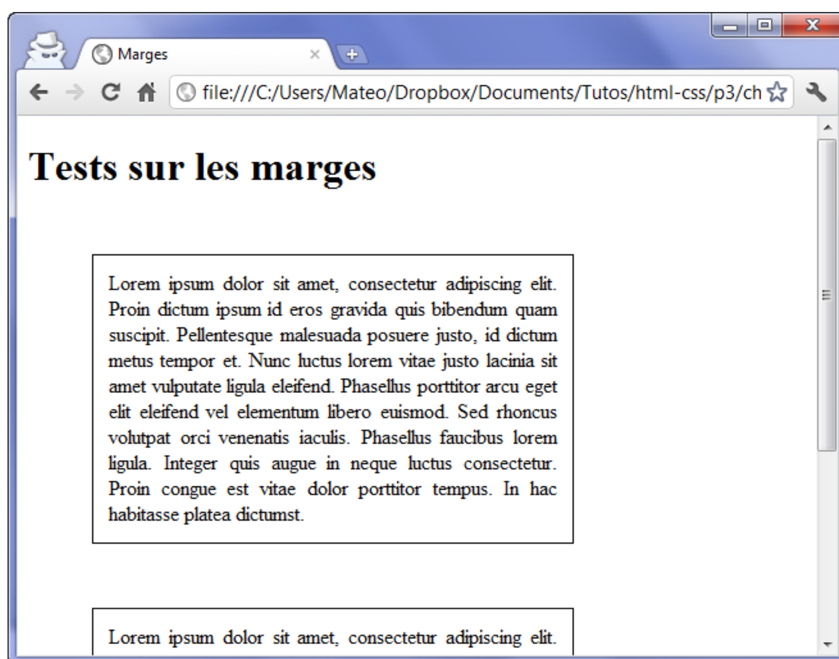


FIGURE 12.7 – Une marge extérieure ajoutée aux paragraphes



Mais ? ? ? Une marge s'est rajoutée à gauche aussi !

Eh oui, `margin` (comme `padding` d'ailleurs) s'applique aux quatre côtés du bloc. Si vous voulez spécifier des marges différentes en haut, en bas, à gauche et à droite, il va falloir utiliser des propriétés plus précises... Le principe est le même que pour la propriété `border`, vous allez voir !

En haut, à droite, à gauche, en bas... Et on recommence !

L'idéal serait que vous reteniez les termes suivants en anglais :

- *top* : haut ;
- *bottom* : bas ;
- *left* : gauche ;
- *right* : droite.

Ainsi, vous pouvez retrouver toutes les propriétés de tête. Je vais quand même vous faire la liste des propriétés pour `margin` et `padding`, histoire que vous soyez sûrs que vous avez compris le principe.

Voici la liste pour `margin` :

- `margin-top` : marge extérieure en haut ;
- `margin-bottom` : marge extérieure en bas ;
- `margin-left` : marge extérieure à gauche ;
- `margin-right` : marge extérieure à droite.

Et la liste pour `padding` :

- `padding-top` : marge intérieure en haut ;
- `padding-bottom` : marge intérieure en bas ;
- `padding-left` : marge intérieure à gauche ;
- `padding-right` : marge intérieure à droite.



Il y a d'autres façons de spécifier les marges avec les propriétés `margin` et `padding`. Par exemple : `margin: 2px 0 3px 1px;` signifie « 2 px de marge en haut, 0 px à droite (le px est facultatif dans ce cas), 3 px en bas, 1 px à gauche ». Autre notation raccourcie : `margin: 2px 1px;` signifie « 2 px de marge en haut et en bas, 1 px de marge à gauche et à droite ».

Centrer des blocs

Il est tout à fait possible de centrer des blocs. C'est même très pratique pour réaliser un design centré quand on ne connaît pas la résolution du visiteur.

Pour centrer, il faut respecter les règles suivantes :

- donnez une largeur au bloc (avec la propriété `width`) ;
- indiquez que vous voulez des marges extérieures automatiques, comme ceci : `margin: auto;`.

Essayons cette technique sur nos petits paragraphes (lignes 3 et 4) :

```
1 p
2 {
3     width: 350px; /* On a indiqué une largeur (obligatoire) */
4     margin: auto; /* On peut donc demander à ce que le bloc
          soit centré avec auto */
5     border: 1px solid black;
6     text-align: justify;
7     padding: 12px;
8     margin-bottom: 20px;
9 }
```

Et voici le résultat à la figure 12.8.

Ainsi, le navigateur centre automatiquement nos paragraphes !

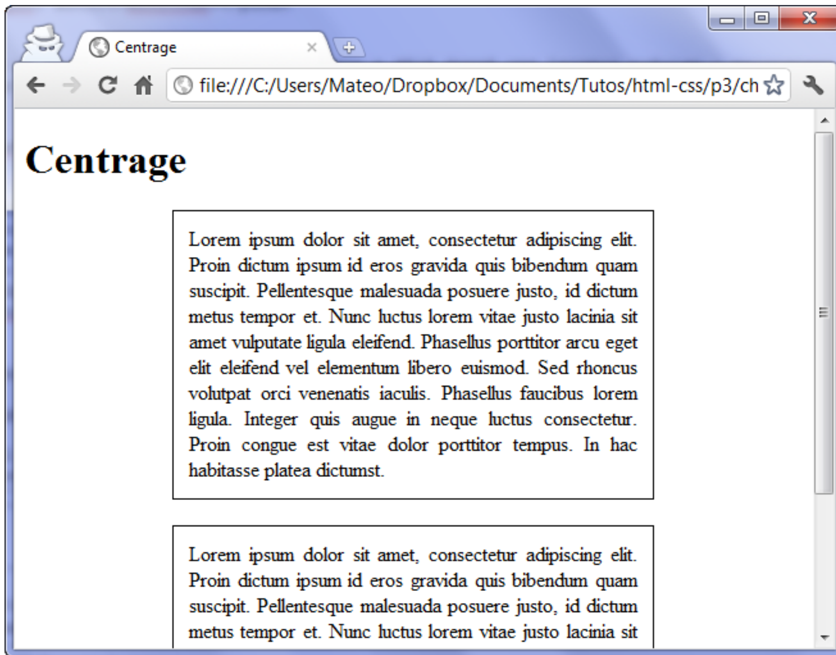


FIGURE 12.8 – Centrage des paragraphes



Il n'est cependant pas possible de centrer verticalement un bloc avec cette technique. Seul le centrage horizontal est permis.

Quand ça dépasse...

Lorsqu'on commence à définir des dimensions précises pour nos blocs, comme on vient de le faire, il arrive qu'ils deviennent trop petits pour le texte qu'ils contiennent.

Les propriétés CSS que nous allons voir ici ont justement été créées pour contrôler les dépassements... et décider quoi faire si jamais cela devait arriver.

`overflow` : couper un bloc

Supposons que vous ayez un long paragraphe et que vous vouliez (pour une raison qui ne regarde que vous) qu'il fasse 250 px de large et 110 px de haut. Ajoutons-lui une bordure et remplissons-le de texte... à ras-bord (figure 12.9) :

```
p
{
    width: 250px;
```

```
    height: 110px;
    text-align: justify;
    border: 1px solid black;
}
```

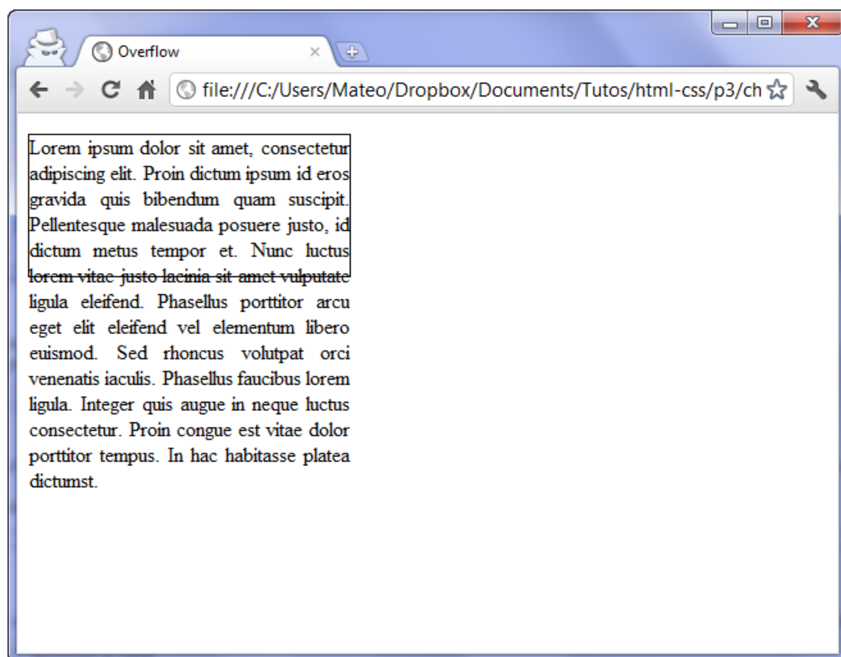


Figure 12.9 – Le texte dépasse du bloc de paragraphe



Horreur ! Le texte dépasse des limites du paragraphe !

Eh oui ! Vous avez demandé des dimensions précises, vous les avez eues ! Mais. . . le texte ne tient pas à l'intérieur d'un si petit bloc.

Si vous voulez que le texte ne dépasse pas des limites du paragraphe, il va falloir utiliser la propriété `overflow`. Voici les valeurs qu'elle peut accepter :

- `visible` (par défaut) : si le texte dépasse les limites de taille, il reste visible et sort volontairement du bloc.
- `hidden` : si le texte dépasse les limites, il sera tout simplement coupé. On ne pourra pas voir tout le texte.
- `scroll` : là encore, le texte sera coupé s'il dépasse les limites. Sauf que cette fois, le navigateur mettra en place des barres de défilement pour qu'on puisse lire l'ensemble du texte. C'est un peu comme un cadre à l'intérieur de la page.
- `auto` : c'est le mode « pilote automatique ». En gros, c'est le navigateur qui décide

de mettre ou non des barres de défilement (il n'en mettra que si c'est nécessaire). C'est la valeur que je conseille d'utiliser le plus souvent.

Avec `overflow: hidden;` le texte est donc coupé (on ne peut pas voir la suite), comme sur la figure 12.10.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin dictum ipsum id eros gravida quis bibendum quam suscipit. Pellentesque malesuada posuere justo, id dictum metus tempor et. Nunc luctus

FIGURE 12.10 – Le texte est coupé aux limites du paragraphe

Essayons maintenant `overflow: auto;` avec le code CSS suivant (résultat à la figure 12.11) :

```
p
{
    width: 250px;
    height: 110px;
    text-align: justify;
    border: 1px solid black;
    overflow: auto;
}
```

Eurêka ! Des barres de défilement nous permettent maintenant de consulter le contenu qui n'était pas visible.



Il existe une ancienne balise HTML, `<iframe>`, qui donne à peu près le même résultat. Cependant, l'usage de cette balise est déconseillé aujourd'hui. Elle permet de charger tout le contenu d'une autre page HTML au sein de votre page.

`word-wrap` : couper les textes trop larges

Si vous devez placer un mot très long dans un bloc, qui ne tient pas dans la largeur, vous allez adorer `word-wrap`. Cette propriété permet de forcer la césure des très longs mots (généralement des adresses un peu longues).

La figure 12.12 représente ce que l'on peut avoir quand on écrit une URL un peu longue dans un bloc.

L'ordinateur ne sait pas « couper » l'adresse car il n'y a ni espace, ni tiret. Il ne sait pas faire la césure.

Avec le code suivant, la césure sera forcée dès que le texte risque de dépasser (figure 12.13).

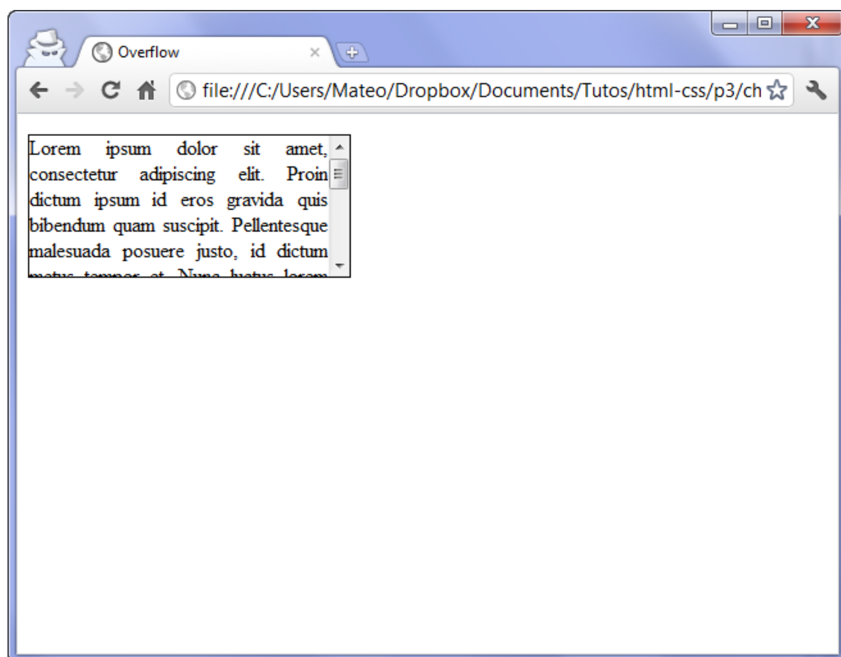


Figure 12.11 – Des barres de défilement sont ajoutées au paragraphe

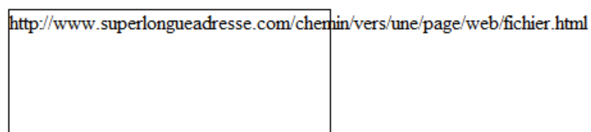


Figure 12.12 – Le texte déborde en largeur

```
p
{
    word-wrap: break-word;
}
```

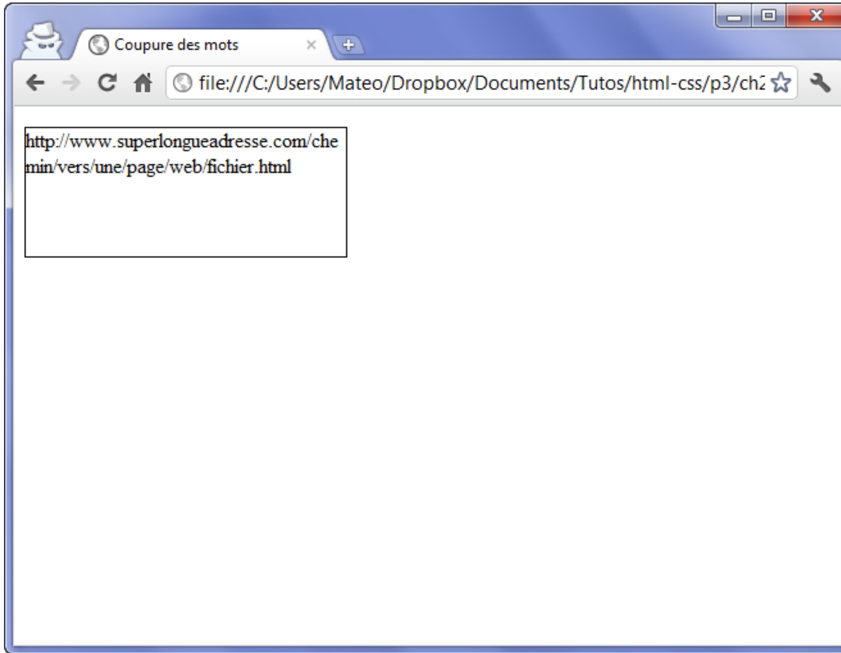


Figure 12.13 – Le texte est coupé pour ne pas déborder



Je conseille d'utiliser cette fonctionnalité dès qu'un bloc est susceptible de contenir du texte saisi par des utilisateurs (par exemple sur les forums de votre futur site). Sans cette astuce, on peut « casser » facilement le design d'un site (en écrivant par exemple une longue suite de « aaaaaaaaaa »).

En résumé

- On distingue deux principaux types de balises en HTML :
  - Le type block (`<p>`, `<h1>`...) : ces balises créent un retour à la ligne et occupent par défaut toute la largeur disponible. Elles se suivent de haut en bas.
  - Le type inline (`<a>`, `<strong>`...) : ces balises délimitent du texte au milieu d'une ligne. Elles se suivent de gauche à droite.
- On peut modifier la taille d'une balise de type block avec les propriétés CSS `width` (largeur) et `height` (hauteur).

- On peut définir des minima et maxima autorisés pour la largeur et la hauteur : `min-width`, `max-width`, `min-height`, `max-height`.
- Les éléments de la page disposent chacun de marges intérieures (`padding`) et extérieures (`margin`).
- S'il y a trop de texte à l'intérieur d'un bloc de dimensions fixes, il y a un risque de débordement. Dans ce cas, il peut être judicieux de rajouter des barres de défilement avec la propriété `overflow` ou de forcer la césure avec `word-wrap`.

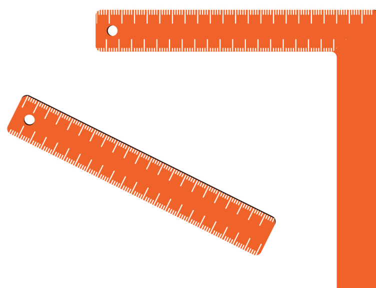
Chapitre 13

Le positionnement en CSS

Difficulté :

Voici venu le moment tant attendu : nous allons apprendre à modifier la position des éléments sur notre page. La théorie que nous allons voir ici nous sera indispensable au prochain chapitre, dans lequel nous réaliserons ensemble, pas à pas, le design de notre premier site !

Vous allez voir, il existe plusieurs techniques permettant d'effectuer la mise en page de son site. Chacune a ses avantages et ses défauts, ce sera à vous de sélectionner celle qui vous semble la meilleure selon votre cas.



Le positionnement flottant



La technique présentée ici l'est à titre d'information. Elle est utilisée par la majorité des sites à l'heure actuelle mais comporte un certain nombre de défauts. Une meilleure technique, le positionnement `inline-block`, sera présentée un peu plus loin : je vous encourage à l'utiliser autant que possible.

Vous vous souvenez de la propriété `float` ? Nous l'avons utilisée pour faire flotter une image autour du texte (figure 13.1).

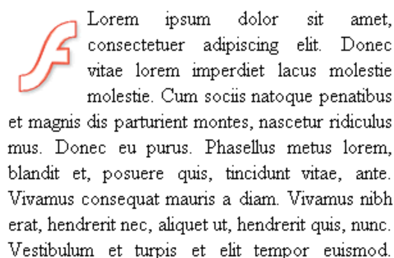


FIGURE 13.1 – L'image flotte autour du texte grâce à la propriété `float`

Il se trouve que cette propriété est aujourd'hui utilisée par la majorité des sites web pour... faire la mise en page ! En effet, si on veut placer son menu à gauche et le contenu de sa page à droite, c'est *a priori* un bon moyen. Je dis bien *a priori* car, à la base, cette propriété n'a pas été conçue pour faire la mise en page et nous allons voir qu'elle a quelques petits défauts.

Reprenons le code HTML structuré que nous avons réalisé il y a quelques chapitres :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Zozor - Le Site Web</title>
    </head>

    <body>
        <header>
            <h1>Zozor</h1>
            <h2>Carnets de voyage</h2>
        </header>

        <nav>
            <ul>
                <li><a href="#">Accueil</a></li>
                <li><a href="#">Blog</a></li>
                <li><a href="#">CV</a></li>
            </ul>
```

```
        </nav>

        <section>
            <aside>
                <h1>À propos de l'auteur</h1>
                <p>C'est moi, Zozor ! Je suis né un 23 novembre 2005.</p>
            </aside>
            <article>
                <h1>Je suis un grand voyageur</h1>
                <p>Bla bla bla bla (texte de l'article)</p>
            </article>
        </section>

        <footer>
            <p>Copyright Zozor - Tous droits réservés

            <a href="#">Me contacter !</a></p>
        </footer>

    </body>
</html>
```


Copier ce code
Code web : 829837

Je rappelle que, sans CSS, la mise en page ressemble à la figure 13.2.

Nous allons essayer de placer le menu à gauche et le reste du texte à droite. Pour cela, nous allons faire flotter le menu à gauche et laisser le reste du texte se placer à sa droite.

Nous voulons que le menu occupe 150 pixels de large. Nous allons aussi rajouter une bordure noire autour du menu et une bordure bleue autour du corps (à la balise `<section>`) pour bien les distinguer :

```
nav
{
    float: left;
    width: 150px;
    border: 1px solid black;
}

section
{
    border: 1px solid blue;
}
```

Voici le résultat à la figure 13.3. Ce n'est pas encore tout à fait cela.

Il y a deux défauts (mis à part le fait que c'est encore bien moche) :

– Le texte du corps de la page touche la bordure du menu. Il manque une petite



FIGURE 13.2 – Une page HTML sans CSS



FIGURE 13.3 – Le menu est bien positionné mais collé au texte

marge...
– Plus embêtant encore : la suite du texte passe... sous le menu !

On veut bien que le pied de page (« Copyright Zozor ») soit placé en bas sous le menu mais, par contre, on aimerait que tout le corps de page soit constitué d'un seul bloc placé à droite.

Pour résoudre ces deux problèmes d'un seul coup, il faut ajouter une marge extérieure à gauche de notre `<section>`, marge qui doit être par ailleurs *supérieure* à la largeur du menu. Si notre menu fait 150 px, nous allons par exemple donner une marge extérieure gauche de 170 px à notre section de page (figure 13.4), ici à la ligne 10.

```
1  nav
2  {
3      float: left;
4      width: 150px;
5      border: 1px solid black;
6  }
7
8  section
9  {
10     margin-left: 170px;
11     border: 1px solid blue;
12 }
```



FIGURE 13.4 – Le corps de page est bien aligné à droite du menu

Voilà, le contenu de la page est maintenant correctement aligné.



À l'inverse, vous pouvez aussi préférer qu'un élément se place obligatoirement sous le menu. Dans ce cas, il faudra utiliser... `clear: both;`, que nous avions déjà découvert, qui oblige la suite du texte à se positionner sous l'élément flottant.

Transformez vos éléments avec `display`

Je vais vous apprendre ici à modifier les lois du CSS (brrr...). Accrochez-vous !

Il existe en CSS une propriété très puissante : `display`. Elle est capable de *transformer* n'importe quel élément de votre page d'un type vers un autre. Avec cette propriété magique, je peux par exemple imposer à mes liens (originellement de type inline) d'apparaître sous forme de blocs :

```
a
{
    display: block;
}
```

À ce moment-là, les liens vont se positionner les uns en-dessous des autres (comme des blocs normaux) et il devient possible de modifier leurs dimensions !

Voici quelques-unes des principales valeurs que peut prendre la propriété `display` en CSS (il y en a encore d'autres) :

Valeur	Exemples	Description
`inline`	`<a>`, `<em>`, `<span>`...	Eléments d'une ligne. Se placent les uns à côté des autres.
`block`	`<p>`, `<div>`, `<section>`...	Eléments en forme de blocs. Se placent les uns en-dessous des autres et peuvent être redimensionnés.
`inline-block`	`<select>`, `<input>`	Eléments positionnés les uns à côté des autres (comme les inlines) mais qui peuvent être redimensionnés (comme les blocs).
`none`	`<head>`	Eléments non affichés.

On peut donc décider de masquer complètement un élément de la page avec cette propriété. Par exemple, si je veux masquer les éléments qui ont la classe « secret », je vais écrire :

```
.secret
{
    display: none;
}
```



Pour faire apparaître ces éléments par la suite, vous devrez faire appel à JavaScript. Certains sites web utilisent cette technique pour, par défaut, masquer les sous-menus qui ne s'affichent que lorsqu'on parcourt les menus.



Et quel est ce nouveau type bizarre, `inline-block` ? C'est un mélange ?

Oui, ce type d'élément est en fait une combinaison des inlines et des blocs. C'est un peu le meilleur des deux mondes : les éléments s'affichent côte à côte et peuvent être redimensionnés.

Peu de balises sont affichées comme cela par défaut, c'est surtout le cas des éléments de formulaire (comme `<input>`) que nous découvrirons un peu plus tard. Par contre, avec la propriété `display`, nous allons pouvoir transformer d'autres balises en `inline-block`, ce qui va nous aider à réaliser notre design.

Le positionnement inline-block

Les manipulations que demande le positionnement flottant se révèlent parfois un peu délicates sur des sites complexes. Dès qu'il y a un peu plus qu'un simple menu à mettre en page, on risque d'avoir à recourir à des `clear: both;` qui complexifient rapidement le code de la page.

Si le positionnement flottant reste, de loin, le mode de positionnement le plus utilisé sur le Web à l'heure actuelle, d'autres techniques existent et bien peu de webmasters le savent. L'une d'elles, étonnamment puissante, est passée sous le nez des concepteurs de sites web alors qu'elle existe depuis CSS 2.1, c'est-à-dire depuis plus de dix ans ! Elle consiste à transformer vos éléments en `inline-block` avec la propriété `display`.

Quelques petits rappels sur les éléments de type inline-block :

- Ils se positionnent les uns à côté des autres (exactement ce qu'on veut pour placer notre menu et le corps de notre page !).
- On peut leur donner des dimensions précises (là encore, exactement ce qu'on veut !).

Nous allons transformer en inline-block les deux éléments que nous voulons placer côte à côte : le menu de navigation et la section du centre de la page.

```
nav
{
    display: inline-block;
    width: 150px;
    border: 1px solid black;
}

section
{
    display: inline-block;
    border: 1px solid blue;
}
```

Ce qui nous donne comme résultat la figure 13.5.



FIGURE 13.5 – Le menu et le corps sont côte à côte... mais positionnés en bas !

Argh ! Ce n'est pas tout à fait ce qu'on voulait. Et en fait, c'est normal : les éléments inline-block se positionnent sur une même ligne de base (appelée **baseline**), en bas.

Heureusement, le fait d'avoir transformé les éléments en inline-block nous permet d'utiliser une nouvelle propriété, normalement réservée aux tableaux : `vertical-align`. Cette propriété permet de modifier l'alignement vertical des éléments. Voici quelques-unes des valeurs possibles pour cette propriété :

- `baseline` : aligne de la base de l'élément avec celle de l'élément parent (par défaut) ;
- `top` : aligne en haut ;
- `middle` : centre verticalement ;
- `bottom` : aligne en bas ;
- (valeur en px ou %) : aligne à une certaine distance de la ligne de base (baseline).

Il ne nous reste plus qu'à aligner nos éléments en haut (lignes 6 et 13), et le tour est joué (figure 13.6) !

```
1 nav
2 {
3     display: inline-block;
4     width: 150px;
5     border: 1px solid black;
6     vertical-align: top;
7 }
8 
9 section
```

```
{
    display: inline-block;
    border: 1px solid blue;
    vertical-align: top;
}
```



Figure 13.6 – Le menu et le corps sont alignés en haut et côte à côte



Vous noterez que le corps (la `<section>`) ne prend pas toute la largeur. En effet, ce n'est plus un bloc ! La section occupe seulement la place dont elle a besoin. Si cela ne vous convient pas pour votre design, modifiez la taille de la section avec `width`.

Et voilà ! Pas besoin de s'embêter avec les marges, aucun risque que le texte passe sous le menu… Bref, c'est parfait !

… Quoi ? Pardon, on me signale dans l'oreillette qu'un certain navigateur vient jouer les trouble-fête…

`inline-block` et compatibilité Internet Explorer

Le positionnement `inline-block` est parfaitement compris par Internet Explorer à partir de IE8.

Pour les anciennes versions, IE6 et IE7 en particulier, le positionnement inline-block fonctionne… mais uniquement sur les éléments qui étaient des inline à la base !

On peut heureusement régler ce problème avec une petite « bidouille » : si le navigateur est IE6 ou IE7, on transforme l'élément en inline et on change son comportement en lui donnant le *layout* (en lui affectant la propriété `zoom: 1;`). L'élément pourra alors être redimensionné.



Le **layout** est un mode d'affichage un peu particulier d'Internet Explorer, qui peut être la source de nombreux bugs. Comme cela devient rapidement très technique, je passe volontairement là-dessus et je vous invite, si le sujet vous intéresse, à consulter l'article concernant `HasLayout` sur le site `Alsacréations`.

▷ Lire l'article
Code web : 640405

Pour commencer, il va falloir créer une seconde feuille de style CSS spéciale pour Internet Explorer 6 et 7. On va utiliser pour cela un **commentaire conditionnel**, que nous avions déjà aperçu auparavant, qui ne sera lu que par IE6 et IE7 :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <!--[if lte IE 7]>
        <link rel="stylesheet" href="style_ie.css" />
        <![endif]-->

        <title>Zozor - Le Site Web</title>
    </head>
```

Si le navigateur est IE6 ou IE7, une seconde feuille de style CSS, `style_ie.css`, sera chargée et viendra s'ajouter à la première. Dedans, nous ferons toutes nos « bidouilles » pour régler les problèmes liés aux vieilles versions d'Internet Explorer, afin que notre site s'affiche correctement chez tout le monde, même chez ceux qui vivent encore à la préhistoire ;-) .

Dans `style_ie.css`, rajoutez simplement le code suivant :

```
nav, section
{
    display: inline;
    zoom: 1;
}
```

Voilà, c'est tout ! Dans votre feuille de style classique (`style.css`), continuez à utiliser `inline-block` comme avant. Ceux qui ont IE6 ou IE7 liront *en plus* ce code un peu spécial, qui leur permet d'obtenir le même résultat visuel !

Les positionnements absolu, fixe et relatif

Il existe d'autres techniques un peu particulières permettant de positionner avec précision des éléments sur la page :

- **Le positionnement absolu** : il nous permet de placer un élément n'importe où sur la page (en haut à gauche, en bas à droite, tout au centre, etc.).
- **Le positionnement fixe** : identique au positionnement absolu mais, cette fois, l'élément reste toujours visible, même si on descend plus bas dans la page. C'est un peu le même principe que `background-attachment: fixed;` (si vous vous en souvenez encore).
- **Le positionnement relatif** : permet de décaler l'élément par rapport à sa position normale.



Comme pour les flottants, les positionnements absolu, fixé et relatif fonctionnent aussi sur des balises de type inline. Toutefois, vous verrez qu'on l'utilise bien plus souvent sur des balises block que sur des balises inline.

Il faut d'abord faire son choix entre les trois modes de positionnement disponibles. Pour cela, on utilise la propriété CSS `position` à laquelle on donne une de ces valeurs :

- `absolute` : positionnement absolu ;
- `fixed` : positionnement fixe ;
- `relative` : positionnement relatif.

Nous allons étudier un à un chacun de ces positionnements.

Le positionnement absolu

Le positionnement absolu permet de placer un élément (réellement) n'importe où sur la page. Pour effectuer un positionnement absolu, on doit écrire :

```
element
{
    position: absolute;
}
```

Mais cela ne suffit pas ! On a dit qu'on voulait un positionnement absolu, mais encore faut-il dire *où l'on veut que le bloc soit positionné sur la page*. Pour ce faire, on va utiliser quatre propriétés CSS :

- `left` : position par rapport à la gauche de la page ;
- `right` : position par rapport à la droite de la page ;
- `top` : position par rapport au haut de la page ;
- `bottom` : position par rapport au bas de la page.

On peut leur donner une valeur en pixels, comme 14px, ou bien une valeur en pourcentage, comme 50%.

Si ce n'est pas très clair pour certains d'entre vous, la figure 13.7 devrait vous aider à comprendre.

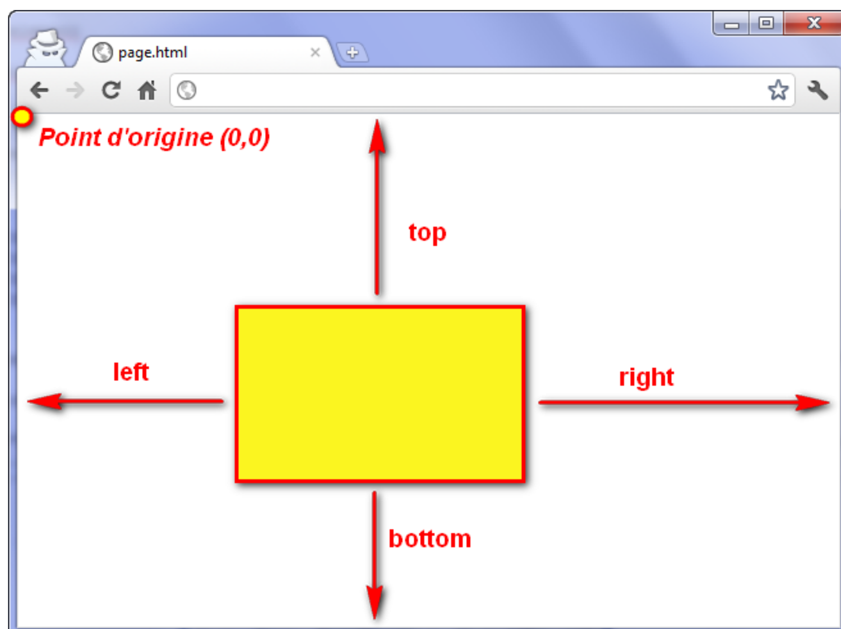


FIGURE 13.7 – Positionnement absolu de l'élément sur la page

Avec cela, vous devriez être capables de positionner correctement votre bloc.

Il faut donc utiliser la propriété `position` et au moins une des quatre propriétés ci-dessus (`top`, `left`, `right` ou `bottom`). Si on écrit par exemple :

```
element
{
    position: absolute;
    right: 0px;
    bottom: 0px;
}
```

... cela signifie que le bloc doit être positionné tout en bas à droite (0 pixel par rapport à la droite de la page, 0 par rapport au bas de la page).

Si on essaye de placer notre bloc `<nav>` en bas à droite de la page, on obtient le même résultat qu'à la figure 13.8.

On peut bien entendu ajouter une marge intérieure (padding) au menu pour qu'il soit moins collé à sa bordure.

Les éléments positionnés en absolu sont placés par-dessus le reste des éléments de la page ! Par ailleurs, si vous placez deux éléments en absolu vers le même endroit, ils risquent de se chevaucher. Dans ce cas, utilisez la propriété `z-index` pour indiquer quel élément doit apparaître au-dessus des autres.



FIGURE 13.8 – Le menu est positionné en bas à droite de l'écran

```
element
{
    position: absolute;
    right: 0px;
    bottom: 0px;
    z-index: 1;
}
element2
{
    position: absolute;
    right: 30px;
    bottom: 30px;
    z-index: 2;
}
```

L'élément ayant la valeur de `z-index` la plus élevée sera placé par dessus les autres, comme le montre la figure 13.9.



Une petite précision technique qui a son importance : le positionnement absolu ne se fait pas forcément toujours par rapport au coin en haut à gauche de la fenêtre ! Si vous positionnez en absolu un bloc A qui se trouve dans un autre bloc B, lui-même positionné en absolu (ou fixe ou relatif), alors votre bloc A se positionnera par rapport au coin supérieur gauche du bloc B. Faites le test, vous verrez !

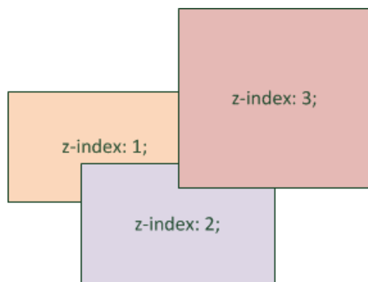


FIGURE 13.9 – Positionnement des éléments absolus

Le positionnement fixe

Le principe est *exactement le même* que pour le positionnement absolu sauf que, cette fois, le bloc reste fixe à sa position, même si on descend plus bas dans la page.

```
element
{
    position: fixed;
    right: 0px;
    bottom: 0px;
}
```

Essayez d'observer le résultat, vous verrez que le menu reste dans le cas présent affiché en bas à droite même si on descend plus bas dans la page (figure 13.10).

Le positionnement relatif

Plus délicat, le positionnement relatif peut vite devenir difficile à utiliser. Ce positionnement permet d'effectuer des « ajustements » : l'élément est décalé par rapport à sa position initiale.

Prenons par exemple un texte important, situé entre deux balises `<strong>`. Pour commencer, je le mets sur fond rouge pour qu'on puisse mieux le repérer :

```
strong
{
   background-color: red; /* Fond rouge */
   color: yellow; /* Texte de couleur jaune */
}
```

Cette fois, le schéma que je vous ai montré tout à l'heure pour les positions absolue et fixe ne marche plus. Pourquoi ? Parce que l'origine a changé : le point de coordonnées (0, 0) ne se trouve plus en haut à gauche de votre fenêtre comme c'était le cas tout à l'heure. Non, cette fois l'origine se trouve en haut à gauche... de la position actuelle de votre élément.

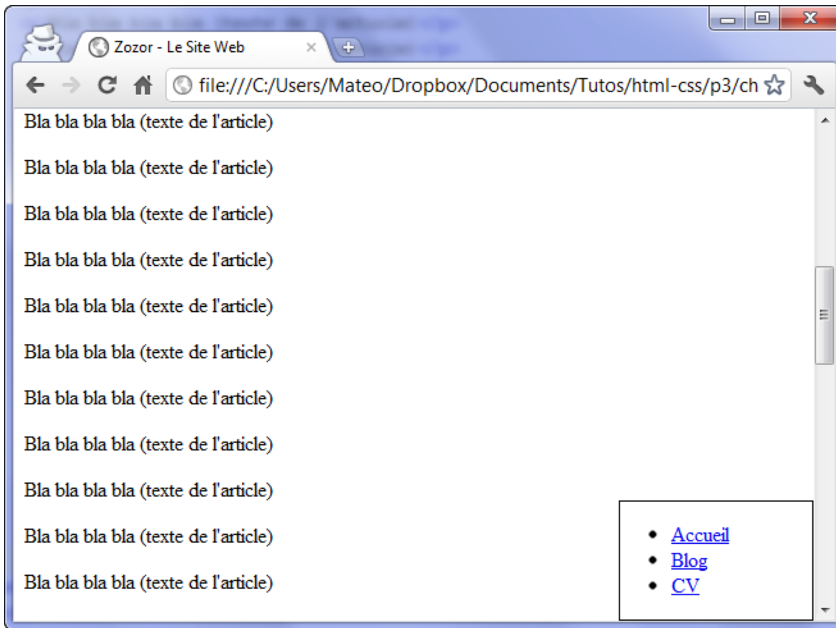


FIGURE 13.10 – Le menu reste affiché en bas à droite en toutes circonstances

Tordu n'est-ce pas ? C'est le principe de la position relative. Le schéma en figure 13.11 devrait vous aider à comprendre où se trouve l'origine des points.

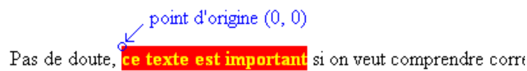


FIGURE 13.11 – Origine de la position relative

Donc, si vous faites un `position: relative;` et que vous appliquez une des propriétés `top`, `left`, `right` ou `bottom`, le texte sur fond rouge va se déplacer par rapport à la position où il se trouve.

Prenons un exemple : je veux que mon texte se décale de 55 pixels vers la droite et de 10 pixels vers le bas. Je vais donc demander à ce qu'il soit décalé de 55 pixels par rapport au « bord gauche » et de 10 pixels par rapport au « bord haut » (lignes 6 à 8) :

```
1 strong
2 {
3     background-color: red;
4     color: yellow;
5 
6     position: relative;
7     left: 55px;
8     top: 10px;
9 }
```

Le texte est alors décalé par rapport à sa position initiale, comme illustré à la figure 13.12.



FIGURE 13.12 – Le texte est décalé

En résumé

- La mise en page d'un site web s'effectue en CSS. Plusieurs techniques sont à notre disposition.
- Le positionnement flottant (avec la propriété `float`) est l'un des plus utilisés à l'heure actuelle. Il permet par exemple de placer un menu à gauche ou à droite de la page. Néanmoins, cette propriété n'a pas été initialement conçue pour cela et il est préférable, si possible, d'éviter cette technique.
- Le positionnement `inline-block` consiste à affecter un type `inline-block` à nos éléments grâce à la propriété `display`. Ils se comporteront comme des inlines (placement de gauche à droite) mais pourront être redimensionnés comme des blocs (avec `width` et `height`). Cette technique est à préférer au positionnement flottant.
- Le positionnement absolu permet de placer un élément où l'on souhaite sur la page, au pixel près.
- Le positionnement fixe est identique au positionnement absolu mais l'élément restera toujours visible même si on descend plus bas dans la page.
- Le positionnement relatif permet de décaler un bloc par rapport à sa position normale.
- Un élément A positionné en absolu à l'intérieur d'un autre élément B (lui-même positionné en absolu, fixe ou relatif) se positionnera par rapport à l'élément B, et non par rapport au coin en haut à gauche de la page.

Chapitre 14

TP : création d'un site pas à pas

Difficulté :

Enfin , nous y voilà. C'est un chapitre un peu particulier, assez différent de ce que vous avez lu jusqu'à maintenant. En fait, c'est ce que j'appelle un « TP » (Travaux Pratiques). Maintenant, vous ne pouvez plus vous contenter de lire mes chapitres à moitié endormis, vous allez devoir mettre la main à la pâte en même temps que moi.

Vous avez lu beaucoup de théorie jusqu'ici mais vous vous demandez sûrement comment font les webmasters pour créer d'aussi beaux sites. Vous vous dites que vous êtes encore loin d'avoir les connaissances nécessaires pour construire tout un site. . . Eh bien vous vous trompez !



Maquettage du design

Je vois d'ici le tableau. Vous vous dites « Chouette, on va créer un site complet », vous ouvrez votre éditeur de texte (Notepad++ par exemple), et vous me regardez en me demandant « Bon, par quelle ligne de code on commence ? ».

Et là, je dois justement vous arrêter. Prenez un crayon et un papier : il faut d'abord réfléchir à ce que vous voulez créer comme site. De quoi va-t-il parler ? Avez-vous un thème, un objectif ?

Je sais, par expérience, que la plupart d'entre vous « cherche juste à apprendre » pour le moment. Vous n'avez donc peut-être pas encore d'idée précise en tête. Dans ce cas, je vous suggère de créer un site pour vous présenter, pour assurer votre présence sur le Web : ce site parlera de vous, il y aura votre CV, vos futures réalisations et pourquoi pas votre blog.

En ce qui me concerne, dans ce TP, je vais réaliser le site web de notre mascotte Zozor, le célèbre âne du Site du Zéro (figure 14.1). Zozor a décidé de partir en voyage à travers le monde et sa première étape sera... San Francisco ! Il veut donc créer un site web pour qu'on le connaisse et pour qu'on suive son périple à travers le monde.



Figure 14.1 – L'âne Zozor, la mascote du Site du Zéro

La première étape consiste à *maquetter le design*, pour avoir un objectif du site web à réaliser. À partir de là, deux possibilités :

- soit vous êtes des graphistes (ou vous en connaissez un) ayant l'habitude d'imaginer des designs, avec des logiciels comme Photoshop ;
- soit vous n'êtes pas très créatifs, vous manquez d'inspiration et, dans ce cas, vous allez chercher votre inspiration sur des sites web comme `freehtml5templates.com`, qui vous proposent des idées de design et qui peuvent même vous donner le code HTML / CSS tout prêt !

Pour ma part j'ai fait appel à Fan Jiyong, un graphiste, qui m'a proposé le design (qui me plaît beaucoup !) que vous pouvez voir à la figure 14.2.

Cette maquette est en fait une simple image du résultat qu'on veut obtenir. Je demande au graphiste de me fournir les éléments qui vont m'aider à construire le design, c'est-à-dire les codes couleurs utilisés, les images découpées (figure 14.3) ainsi que les polices dont j'aurai besoin.

▷ Télécharger les images et les polices
Code web : 991714

Zozor
Carnets de voyage

ACCUEIL BLOG CV CONTACT

Retour sur mes vacances aux États-Unis... Voir l'article

JE SUIS UN GRAND VOYAGEUR

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris metus massa, luctus in tincidunt a, porttitor ac leo. Maecenas at mi feugiat turpis elementum ornare. Sed tempor rutrum lorem, in vestibulum felis elementum ac. Fusce purus orci, scelerisque ut tincidunt in, dignissim vel augue. Nulla iaculis ultrices sagittis. Nulla vitae neque dignissim enim tempor scelerisque at quis tellus. In hac habitasse platea dictumst. Aenean elit elit, pellentesque nec venenatis ut, convallis eu sem. Mauris eu leo nec arcu volutpat euismod nec eu dolor. Morbi aliquet, mauris quis porttitor dapibus, odio enim viverra eros, quis interdum massa urna at velit. Integer tempor facilisis libero non accumsan. Aliquam diam felis, dapibus sed condimentum quis, molestie vel odio. Maecenas eget ante massa, a sagittis quam. Cras posuere magna ac urna molestie vitae luctus lacus lobortis. Quisque leo neque, vulputate at semper non, varius porta enim.

Praesent sit amet lectus eros, ac pellentesque nisl. Donec consequat magna sed libero condimentum vitae aliquet elit ornare. Nunc at nulla purus. Aliquam sit amet sapien sit amet nisi aliquet rutrum vel nec mi. Mauris ultricies felis egestas mi varius molestie molestie sapien tristique. Cras lacus lacus, rutrum id sagittis sit amet, malesuada nec odio. Nulla consectetur lobortis libero, ac convallis massa consectetur in. Nam facilisis posuere sagittis. Sed a ligula id dui vulputate congue quis at tortor. Nunc pellentesque faucibus felis, eu venenatis massa interdum in. Donec venenatis lacus id tortor vestibulum id accumsan est lobortis. Morbi turpis quam, tincidunt in accumsan quis, ullamcorper quis orci. Quisque nisl magna, egestas eget consectetur non, mollis ac ante. Donec elit felis, blandit at auctor in, lacinia et dolor.

Ut blandit, diam id aliquam volutpat, quam libero euismod neque, ut volutpat nunc ipsum a magna. Donec hendrerit sem in dolor egestas lobortis. Etiam bibendum lobortis interdum. Etiam ac felis vitae neque sodales sodales. Nunc tempus dignissim dapibus. Duis sit amet tellus vitae elit suscipit convallis. Sed et tincidunt velit. Donec congue elementum ante eu consectetur. Morbi lectus mauris, sodales a euismod id, dapibus sollicitudin urna. Sed sagittis sagittis placerat. Etiam at lorem risus. Quisque imperdiet elementum tortor nec viverra. tempus dignissim dapibus. Duis sit amet tellus vitae elit suscipit convallis. Sed et tincidunt velit. Donec congue elementum ante eu consectetur. Morbi lectus mauris, sodales a euismod id, dapibus sollicitudin urna. Sed sagittis sagittis placerat.

À PROPOS DE L'AUTEUR

Laisse-moi le temps de me présenter : je m'appelle ZoZor, je suis né un 23 Novembre 2005.

Bien maigre, n'est-ce pas ? C'est pourquoi, aujourd'hui, j'ai décidé d'écrire ma biographie (ou zBiographie, comme vous voulez !) afin que les zéros sachent enfin qui je suis réellement.

MON DERNIER TWEET

Hii haaaaaan !

le 12 mai à 23h12

MES PHOTOS

MES AMIS

- Pupi le lapin
- Mr Baobab
- Kaiwaii
- Perceval.eu
- Belette
- Le concombre masqué
- Ptit prince
- Mr Fan

FIGURE 14.2 – La maquette du site web que nous allons réaliser

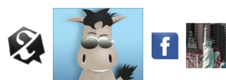


FIGURE 14.3 – Quelques images « découpées » utilisées dans le design

Il ne nous reste plus qu'à réaliser ce site web ! Nous allons procéder en deux temps :

1. Nous allons construire le squelette **HTML** de la page.
2. Puis nous allons le mettre en forme et le mettre en page avec **CSS**.

Allez, au boulot !

Organiser le contenu en HTML

La première chose à faire est de distinguer les principaux blocs sur la maquette. Ces blocs vont constituer le squelette de notre page.

Pour créer ce squelette, nous allons utiliser différentes balises HTML :

- les balises structurantes de HTML5, que nous connaissons : `<header>`, `<section>`, `<nav>`, etc. ;
- la balise universelle `<div>` quand aucune balise structurante ne convient.



Comment je sais quelle balise utiliser moi ?

C'est à vous de décider. De préférence, utilisez une balise qui a du sens (comme les balises structurantes `<header>`, `<section>`, `<nav>`) mais, si aucune balise ne vous semble mieux convenir, optez pour la balise générique `<div>`.

Regardez la figure 14.4 pour voir ce que je vous propose comme structure.



On peut imaginer d'autres façons de faire le découpage, retenez bien que ma proposition n'est pas forcément la seule et unique solution !

Toutes les balises que l'on va utiliser n'apparaissent pas sur cette maquette mais cela vous permet d'avoir une idée de l'imbrication que je propose pour les éléments.

Le HTML n'est pas vraiment la partie complexe de la réalisation du site web. En fait, si vous avez bien compris comment imbriquer des balises, vous ne devriez pas avoir de mal à réaliser un code approchant du mien :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <title>Zozor - Carnets de voyage</title>
    </head>

    <body>
        <div id="bloc_page">
```



Figure 14.4 – La maquette découpée en différentes sections

```
            <header>
                <div id="titre_principal">
                    <img src="images/zozor_logo.png" alt="Logo de Zozor" id="logo" />
                    <h1>Zozor</h1>
                    <h2>Carnets de voyage</h2>
                </div>

                <nav>
                    <ul>
                        <li><a href="#">Accueil</a></li>
                        <li><a href="#">Blog</a></li>
                        <li><a href="#">CV</a></li>
                        <li><a href="#">Contact</a></li>
                    </ul>
                </nav>
            </header>

            <div id="banniere_image">
                <div id="banniere_description">
                    Retour sur mes vacances aux États-Unis...
                    <a href="#" class="bouton_rouge">Voir l'article <img src="images/flecheblanchedroite.png" alt="" /></a>
                </div>
            </div>

            <section>
                <article>
                    <h1><img src="images/ico_epingle.png" alt="Catégorie voyage" class="ico_categorie" />Je suis un grand voyageur</h1>
                    <p>Lorem ipsum dolor sit amet...</p>
                    <p>Vivamus sed libero nec mauris pulvinar facilisis ut non sem...</p>
                    <p>Phasellus ligula massa, congue ac vulputate non, dignissim at augue...</p>
                </article>
                <aside>
                    <h1>À propos de l'auteur</h1>
                    <img src="images/bulle.png" alt="" id="fleche_bulle" />
                    <p id="photo_zozor"><img src="images/zozor_classe.png" alt="Photo de Zozor" /></p>
                    <p>Laisse-moi le temps de me présenter : je m'appelle Zozor, je suis né un 23 novembre 2005.</p>
                    <p>Bien maigre, n'est-ce pas ? C'est pourquoi, aujourd'hui, j'ai décidé d'é
```

```
crire ma biographie (ou zBiographie, comme vous voulez !) afin que les zéros sachent qui je suis réellement.</p>
                <p><img src="images/facebook.png" alt="Facebook" /><img src="images/twitter.png" alt="Twitter" /><img src="images/vimeo.png" alt="Vimeo" /><img src="images/flickr.png" alt="Flickr" /><img src="images/rss.png" alt="RSS" /></p>
            </aside>
        </section>

        <footer>
            <div id="tweet">
                <h1>Mon dernier tweet</h1>
                <p>Hii haaaaaan !</p>
                <p>le 12 mai à 23h12</p>
            </div>
            <div id="mes_photos">
                <h1>Mes photos</h1>
                <p><img src="images/photo1.jpg" alt="Photographie" /><img src="images/photo2.jpg" alt="Photographie" /><img src="images/photo3.jpg" alt="Photographie" /><img src="images/photo4.jpg" alt="Photographie" /></p>
            </div>
            <div id="mes_amis">
                <h1>Mes amis</h1>
                <ul>
                    <li><a href="#">Pupi le lapin</a></li>
                    <li><a href="#">Mr Baobab</a></li>
                    <li><a href="#">Kaiwaii</a></li>
                    <li><a href="#">Perceval.eu</a></li>
                </ul>
                <ul>
                    <li><a href="#">Belette</a></li>
                    <li><a href="#">Le concombre masqué</a></li>
                    <li><a href="#">Ptit prince</a></li>
                    <li><a href="#">Mr Fan</a></li>
                </ul>
            </div>
        </footer>
    </div>
  </body>
</html>
```

▷ Copier ce code
Code web : 244518

Petite particularité : comme vous le voyez, tout le contenu de la page est placé dans une grande balise `<div>` ayant pour `id` `bloc_page` (on l'appelle aussi parfois `main_wrapper` en anglais). Cette balise englobe tout le contenu, ce qui va nous permettre de fixer facilement les dimensions de la page et de centrer notre site à l'écran.

Pour le reste, aucune grosse difficulté à signaler. Notez que je n'ai pas forcément pensé à toutes les balises du premier coup : en réalisant le design en CSS, il m'est parfois apparu qu'il était nécessaire d'englober une partie des balises d'un bloc `<div>` pour m'aider dans la réalisation du design.

Pour le moment, comme vous vous en doutez, le site web n'est pas bien beau (et encore, je suis gentil). Vous pouvez voir le résultat actuel à la figure 14.5.



FIGURE 14.5 – Apparence du site web constitué uniquement du HTML

C'est en CSS que la magie va maintenant opérer.

Mettre en forme en CSS

Les choses se compliquent un peu plus lorsqu'on arrive au CSS. En effet, il faut du travail (et parfois un peu d'astuce) pour obtenir un résultat se rapprochant de la maquette. Je dis bien « se rapprochant » car vous ne pourrez jamais obtenir un résultat identique au pixel près.

Mettez-vous bien cela en tête : le but est d'obtenir le rendu *le plus proche possible*, sans chercher la perfection. Même si vous obtenez selon vous « la perfection » sur un

navigateur, vous pouvez être sûrs qu'il y aura des différences sur un autre navigateur (plus ancien) ou sur une autre machine que la vôtre. Nous allons donc faire au mieux et ce sera déjà du travail, vous verrez.

Pour mettre en forme le design, je vais procéder en plusieurs étapes. Je vais m'occuper des éléments suivants, dans cet ordre :

1. Polices personnalisées.
2. Définition des styles principaux de la page (largeur du site, fond, couleur par défaut du texte).
3. En-tête et liens de navigation.
4. Bannière (représentant le pont de San Francisco).
5. Section principale du corps de page, au centre.
6. Pied de page (*footer*).

Les polices personnalisées

Pour les besoins du design, mon graphiste a utilisé trois polices sur sa maquette :

- `Trebuchet MS` (police courante) ;
- `BallparkWeiner` (police exotique) ;
- `Day Roman` (police exotique).



Vous trouverez ces polices dans le fichier que je vous ai fait télécharger un peu plus haut. Si ce n'est pas encore fait, je vous encourage fortement à le télécharger.

La plupart des ordinateurs sont équipés de `Trebuchet MS` (quoique pas nécessairement tous, on pourrait la faire télécharger). Par contre, les deux autres polices sont un peu originales et ne sont sûrement pas présentes sur les ordinateurs de vos visiteurs. Nous allons les leur faire télécharger.

Comme vous le savez, il faut proposer plusieurs versions de ces polices pour les différents navigateurs. `Dafont` ne propose que le `.ttf` en téléchargement. Par contre, `FontSquirrel` propose un générateur de polices à utiliser en CSS3 avec `@font-face` : vous lui envoyez un `.ttf`, l'outil transforme le fichier dans tous les autres formats nécessaires et vous fournit même le code CSS prêt à l'emploi !

▷ Générateur de polices
Code web : 688318

Je m'en suis servi pour générer les différentes versions des deux polices exotiques que je vais utiliser. Ensuite, dans mon fichier CSS, je rajoute ce code qui m'a été fourni par `FontSquirrel` pour déclarer les nouvelles polices :

```
/* Définition des polices personnalisées */

@font-face
```

```
{
    font-family: 'BallparkWeiner';
    src: url('polices/ballpark.eot');
    src: url('polices/ballpark.eot?#iefix') format('embedded-
        opentype'),
         url('polices/ballpark.woff') format('woff'),
         url('polices/ballpark.ttf') format('truetype'),
         url('polices/ballpark.svg#BallparkWeiner') format('svg
            ');
    font-weight: normal;
    font-style: normal;
}

@font-face
{
    font-family: 'Dayrom';
    src: url('polices/dayrom.eot');
    src: url('polices/dayrom.eot?#iefix') format('embedded-
        opentype'),
         url('polices/dayrom.woff') format('woff'),
         url('polices/dayrom.ttf') format('truetype'),
         url('polices/dayrom.svg#Dayrom') format('svg');
    font-weight: normal;
    font-style: normal;
}
```

En plus de cela, il faut bien entendu mettre à disposition les fichiers des polices. Comme vous le voyez, j'ai créé un sous-dossier `polices` dans lequel j'ai mis les différentes versions de mes polices.

Définition des styles principaux

On peut maintenant s'attaquer à définir quelques styles globaux pour tout le design de notre page. On va définir une image de fond, une police et une couleur de texte par défaut, et surtout on va dimensionner notre page et la centrer à l'écran.

```
/* Eléments principaux de la page */

body
{
    background: url('images/fond_jaune.png');
    font-family: 'Trebuchet MS', Arial, sans-serif;
    color: #181818;
}

#bloc_page
{
    width: 900px;
    margin: auto;
```

```
}

section h1, footer h1, nav a
{
    font-family: Dayrom, serif;
    font-weight: normal;
    text-transform: uppercase;
}
```

Avec `#bloc_page`, le bloc qui englobe toute la page, j'ai fixé les limites à 900 pixels de large. Avec les marges automatiques, le design sera centré.



Si vous souhaitez créer un design qui s'adapte aux dimensions de l'écran du visiteur, définissez une largeur en pourcentage plutôt qu'en pixels.

J'ai utilisé la propriété CSS `text-transform: uppercase;` (que nous n'avons pas vue auparavant) pour faire en sorte que mes titres soient toujours écrits en majuscules. Cette propriété transforme en effet le texte en majuscules (elle peut aussi faire l'inverse avec `lowercase`). Notez qu'on aurait aussi pu écrire les titres directement en majuscules dans le code HTML.

La figure 14.6 vous montre ce qu'on obtient pour le moment avec le code CSS. On est encore loin du résultat final mais on se sent déjà un petit peu plus « chez soi ».

En-tête et liens de navigation

D'après la structure que j'ai proposée, l'en-tête contient aussi les liens de navigation. Commençons par définir l'en-tête et, en particulier, le logo en haut à gauche. Nous verrons ensuite comment mettre en forme les liens de navigation.

L'en-tête

```
/* Header */

header
{
    background: url('images/separateur.png') repeat-x bottom;
}

#titre_principal
{
    display: inline-block;
}

header h1
{
    font-family: 'BallparkWeiner', serif;
```



Figure 14.6 – Le fond et les limites de la page commencent à apparaître

```
    font-size: 2.5em;
    font-weight: normal;
}

#logo, header h1
{
    display: inline-block;
    margin-bottom: 0px;
}

header h2
{
    font-family: Dayrom, serif;
    font-size: 1.1em;
    margin-top: 0px;
    font-weight: normal;
}
```

Nous créons une distinction entre l'en-tête et le corps de page grâce à une image de fond. Les éléments sont positionnés en `inline-block` et nous personnalisons les polices et les dimensions. Rien d'extraordinaire pour le moment.

Les liens de navigation

La mise en forme des liens de navigation est un petit peu plus intéressante. Vous l'avez vu, j'ai créé une liste à puces pour les liens. . . mais une telle liste s'affiche habituellement en hauteur, et non en largeur. Heureusement, cela se change facilement, vous allez voir :

```
/* Navigation */

nav
{
    display: inline-block;
    width: 740px;
    text-align: right;
}

nav ul
{
    list-style-type: none;
}

nav li
{
    display: inline-block;
    margin-right: 15px;
}

nav a
{
    font-size: 1.3em;
```

```
    color: #181818;
    padding-bottom: 3px;
    text-decoration: none;
}

nav a:hover
{
    color: #760001;
    border-bottom: 3px solid #760001;
}
```

La principale nouveauté est la définition CSS `list-style-type: none;`, qui permet de retirer l'image ronde servant de puce. Chaque élément de la liste (`<li>`) est positionné en `inline-block`, ce qui nous permet de placer les liens côte à côte comme nous le souhaitions.

Le reste des définitions ne contient rien d'extraordinaire : des dimensions, des couleurs, des bordures... Autant de choses que vous connaissez déjà. Notez que je ne trouve pas forcément les bonnes valeurs du premier coup, il me faut parfois tâtonner un peu pour trouver une apparence proche de la maquette d'origine.

La figure 14.7 représente le résultat que nous obtenons avec les derniers ajouts de CSS.

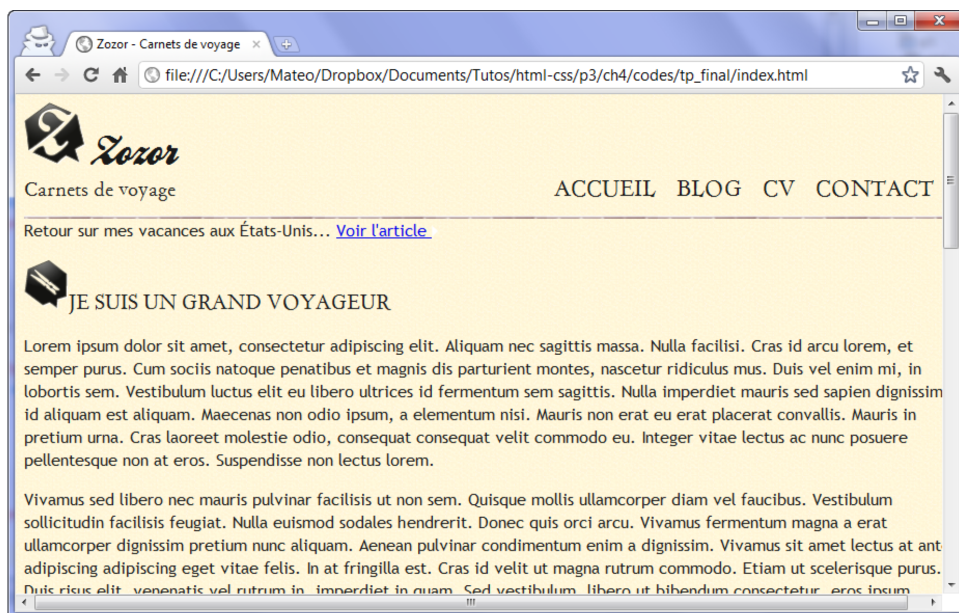


FIGURE 14.7 – L'en-tête est mis en page

La bannière

Bien, passons maintenant à un exercice un peu plus difficile mais très intéressant : la bannière ! Notre maquette comporte une jolie bannière représentant le pont de San Francisco. Cette bannière, sur votre site, peut être amenée à évoluer. Ici, elle peut servir à illustrer, par exemple, le dernier billet de blog de notre ami Zozor, qui vient de visiter San Francisco.

La bannière est intéressante à plus d'un titre :

- elle comporte des angles arrondis ;
- la description est écrite sur un fond légèrement transparent ;
- le bouton « Voir l'article » est réalisé en CSS, avec des angles arrondis ;
- une ombre vient donner du volume à la bannière.

Voici le code que j'ai utilisé pour réaliser toute la bannière :

```
/* Bannière */

#banniere_image
{
    margin-top: 15px;
    height: 200px;
    border-radius: 5px;
    background: url('images/sanfrancisco.jpg') no-repeat;
    position: relative;
    box-shadow: 0px 4px 4px #1c1a19;
    margin-bottom: 25px;
}

#banniere_description
{
    position: absolute;
    bottom: 0;
    border-radius: 0px 0px 5px 5px;
    width: 99.5%;
    height: 33px;
    padding-top: 15px;
    padding-left: 4px;
    background-color: rgb(24,24,24); /* Pour les anciens
        navigateurs */
    background-color: rgba(24,24,24,0.8);
    color: white;
    font-size: 0.8em;
}

.bouton_rouge
{
    display: inline-block;
    height: 25px;
    position: absolute;
    right: 5px;
```

```
    bottom: 5px;
    background: url('images/fond_degraderouge.png') repeat-x;
    border: 1px solid #760001;
    border-radius: 5px;
    font-size: 1.2em;
    text-align: center;
    padding: 3px 8px 0px 8px;
    color: white;
    text-decoration: none;
}

.bouton_rouge img
{
    border: 0;
}
```

Ce code est assez technique et riche en fonctionnalités CSS. C'est peut-être la partie la plus délicate à réaliser dans cette page.

Vous pouvez constater que j'ai choisi d'afficher l'image du pont sous forme d'image de fond dans le bloc `<div>` de la bannière.

J'ai aussi donné une position relative à la bannière, sans utiliser de propriétés pour en modifier le décalage... Pourquoi ? *A priori*, une position relative sans décalage ne sert à rien... Et pourtant, cela m'a été particulièrement utile pour placer le bouton « Voir l'article » en bas à droite de la bannière. En effet, j'ai placé le bouton en absolu à l'intérieur.



Le bouton ne devrait-il pas se placer en bas à droite de la page ?

Non, souvenez-vous ce que je vous avais dit : si un bloc est positionné en absolu dans un autre bloc lui-même positionné en absolu, fixe ou relatif, alors il se positionne *à l'intérieur* de ce bloc. Notre bannière est positionnée en relatif (sans décalage). Comme le bouton est positionné en absolu à l'intérieur, il se place donc en bas à droite de la bannière !

C'est une technique particulièrement utile et puissante dans la réalisation d'un design, souvenez-vous en !

Dernier détail : pour la légende de la bannière, j'ai choisi d'utiliser la transparence avec la notation `RGBa` plutôt que la propriété `opacity`. En effet, `opacity` aurait rendu tout le contenu du bloc transparent, y compris le bouton « Voir l'article » à l'intérieur. J'ai trouvé préférable de rendre transparente seulement la couleur de fond plutôt que tout le bloc.

Le résultat est plutôt sympathique (figure 14.8).

Cela en jette, vous ne trouvez pas ?



FIGURE 14.8 – La bannière est mise en forme



Pour réaliser le dégradé du bouton « Voir l'article », j'ai utilisé une image de fond représentant le dégradé et j'ai répété cette image horizontalement. Sachez qu'il existe une propriété CSS3 `linear-gradient` qui permet de réaliser des dégradés sans avoir à recourir à une image de fond. Son usage étant un peu complexe actuellement, j'ai choisi de ne pas l'utiliser dans cet exemple, mais vous pouvez vous documenter à son sujet si vous le souhaitez !

Le corps

Le corps, au centre de la page, est dans notre cas constitué d'une unique balise `<section>` (mais il pourrait y en avoir plusieurs, bien sûr).

Pas beaucoup de difficultés sur le corps, le positionnement du bloc « À propos de l'auteur » se fait en inline-block. On joue avec les angles arrondis et les ombres, on ajuste un peu les marges et les dimensions du texte, et nous y voilà !

```
/* Corps */

article, aside
{
    display: inline-block;
    vertical-align: top;
    text-align: justify;
}

```

```
article
{
    width: 625px;
    margin-right: 15px;
}

.ico_categorie
{
    vertical-align: middle;
    margin-right: 8px;
}

article p
{
    font-size: 0.8em;
}

aside
{
    position: relative;
    width: 235px;
    background-color: #706b64;
    box-shadow: 0px 2px 5px #1c1a19;
    border-radius: 5px;
    padding: 10px;
    color: white;
    font-size: 0.9em;
}

#fleche_bulle
{
    position: absolute;
    top: 100px;
    left: -12px;
}

#photo_zozor
{
    text-align: center;
}

#photo_zozor img
{
    border: 1px solid #181818;
}

aside img
{
    margin-right: 5px;
}
```

La petite difficulté ici était de réussir à placer la flèche à gauche du bloc `<aside>` « À propos de l'auteur » pour donner l'effet d'une bulle. Là encore, notre meilleur ami est le positionnement absolu. La technique est la même : je positionne le bloc `<aside>` en relatif (sans effectuer de décalage), ce qui me permet ensuite de positionner l'image de la flèche en absolu par rapport au bloc `<aside>` (et non par rapport à la page entière). En jouant sur le décalage de l'image, je peux la placer avec précision où je veux, au pixel près (figure 14.9) !



FIGURE 14.9 – Le corps de la page est mis en forme

Le pied de page

Il ne nous reste plus que le pied de page à mettre en forme. Celui-ci est constitué de trois sous-blocs que j'ai matérialisés par des `<div>` auxquels j'ai donné des `id` pour mieux les repérer. Ces blocs sont positionnés en `inline-block` les uns à côté des autres.

```css
/* Footer */

footer
{
    background: url('images/ico_top.png') no-repeat top center
        , url('images/separateur.png') repeat-x top, url('images
        /ombre.png') repeat-x top;
    padding-top: 25px;
}

```

```css
footer p, footer ul
{
    font-size: 0.8em;
}

footer h1
{
    font-size: 1.1em;
}

#tweet, #mes_photos, #mes_amis
{
    display: inline-block;
    vertical-align: top;
}

#tweet
{
    width: 28%;
}

#mes_photos
{
    width: 35%;
}

#mes_amis
{
    width: 31%;
}

#mes_photos img
{
    border: 1px solid #181818;
    margin-right: 2px;
}

#mes_amis ul
{
    display: inline-block;
    vertical-align: top;
    margin-top: 0;
    width: 48%;
    list-style-image: url('images/ico_liensexterne.png');
    padding-left: 2px;
}

#mes_amis a
{
    text-decoration: none;
```

```
    color: #760001;
}
```

Deux petites particularités à signaler sur le pied de page :

- J'ai utilisé la fonctionnalité des images de fond multiples de CSS3, ce qui m'a permis de réaliser le séparateur entre le corps et le pied de page. Il est constitué de trois images : le séparateur, la petite flèche vers le haut et un léger dégradé.
- J'ai modifié la puce de la liste « Mes amis », en bas à droite, avec la propriété `list-style-image` qui m'a permis d'utiliser une image personnalisée plutôt que les puces standard. Il existe de nombreuses propriétés CSS spécifiques comme celle-ci et nous ne pouvons pas toutes les voir une par une dans ce cours mais, maintenant que vous êtes des habitués du CSS, vous n'aurez aucun mal à apprendre à les utiliser simplement en lisant l'annexe fournissant la liste des principales propriétés CSS (page 291).

Et voilà, notre design est terminé (figure 14.10) !



FIGURE 14.10 – Le pied de page est mis en forme

Ah, vous pensez en avoir fini ? Il reste hélas encore un peu de travail : il faut tester notre site sur différents navigateurs. Idéalement, il vaut mieux le faire au fur et à mesure de la mise en place du design. En particulier, les anciennes versions d'Internet Explorer (IE6 à IE8) méritent qu'on s'y attarde car le résultat n'est pas forcément celui auquel on s'attendait. . .

Assurer la compatibilité avec IE

Depuis Internet Explorer 9 (IE9), nous n'avons plus vraiment de raisons de nous plaindre du légendaire retard d'Internet Explorer dans la gestion du CSS. Voyez vous-mêmes le résultat (figure 14.11), il est très bon sur ce navigateur sans aucune adaptation nécessaire.



Figure 14.11 – Le site sous IE9 : aucun problème à signaler !

Par contre, vous risquez d'attraper quelques cheveux blancs en regardant le résultat sur les anciennes versions d'Internet Explorer.



Comment je fais pour voir le résultat sous IE6 à IE8, si je suis équipé d'IE9 ?

Je vous avais parlé d'IETester, un outil pratique mais instable (il plante souvent). Vous pouvez l'essayer pour tester votre site sur les anciennes versions d'IE.

Il y a cependant une solution plus stable et plus rapide : appuyez sur la touche F12 de votre clavier lorsque vous êtes sous IE9 et une barre dédiée aux développeurs web apparaît. Là, un menu vous permet de changer le comportement d'IE (figure 14.12) afin de simuler les anciennes versions (à partir de IE7).

Là, en général, il arrive qu'on prenne très peur. La figure 14.13 vous montre ce qu'on obtient en « mode IE7 ».

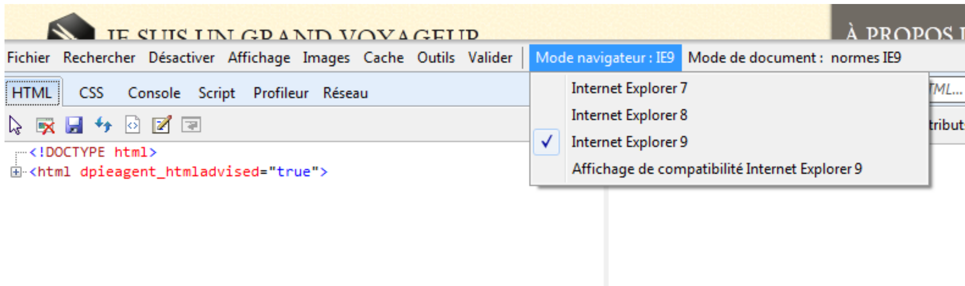


Figure 14.12 – Changement du moteur de rendu sous IE



Figure 14.13 – Le site sous IE7 : rien ne s'affiche correctement !

Avant de préparer votre valise pour aller vivre en Patagonie loin de ce monde cruel, laissez-moi vous redonner le moral avec cette phrase rassurante : *tout problème a une solution* (répétez cela autant de fois que nécessaire).

En fait, notre site ne rencontre que deux problèmes principaux sur les anciennes versions d'IE :

- Le positionnement `inline-block` n'est pas bien géré sous IE6 et IE7, ce qui fait que la plupart de nos positionnements ne fonctionnent pas pour le moment... Mais nous avons vu qu'une astuce permet de régler le problème sans trop d'efforts !
- Les balises structurantes de HTML5 (<header>, <nav>, <aside>...) ne sont pas gérées sous IE6, IE7 et IE8, ce qui pose de gros problèmes d'affichage... Mais, là encore, un petit script ajouté au début de votre code HTML permet de régler le problème !

Par contre, il faudra faire une croix sur certaines fonctionnalités plus récentes de CSS3 qui ne sont pas gérées sur ces vieilles versions :

- les coins arrondis ;
- les images de fond multiples ;
- la transparence ;
- les ombres.

Étant donné que ce sont des fonctionnalités liées à l'apparence, nous ne chercherons pas à les faire fonctionner sur les anciennes versions d'IE. Si toutefois vous y tenez, sachez que, là encore, des scripts existent et permettent d'émuler la plupart de ces fonctionnalités manquantes, mais cela vous demandera pas mal de travail supplémentaire et votre site risque d'être plus lent sur ces navigateurs. Du moment que le site reste lisible sur les anciennes versions d'IE, je vous recommande de ne pas vous préoccuper trop de ces problèmes.



Lorsqu'on accepte que son site soit « un peu moins beau » sur les anciennes versions des navigateurs, on dit qu'on fait une **dégradation gracieuse**. Cela veut dire qu'on ne cherche pas à obtenir tous les effets, mais on s'efforce d'avoir un site qui reste quand même lisible sur les vieux navigateurs.

Faire fonctionner les balises structurantes de HTML5

Nous l'avons vu, il suffit d'ajouter un simple bout de code JavaScript dans l'en-tête de son site et le tour est joué (lignes 6 à 8) :

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="utf-8" />
5         <link rel="stylesheet" href="style.css" />
6         <!--[if lt IE 9]>
7         <script src="http://html5shiv.googlecode.com/svn/trunk/
              html5.js"></script>
```

```
        <![endif]-->
        <title>Zozor - Carnets de voyage</title>
    </head>
```



Le fichier JavaScript sera téléchargé depuis les serveurs de Google : c'est une technique rapide qui vous évite d'avoir à gérer vous-mêmes le fichier. Si vous le souhaitez, vous pouvez aussi en récupérer une copie et la placer à côté des fichiers de votre site.

Régler le positionnement inline-block

Pour gérer le positionnement `inline-block`, nous avons vu qu'il était nécessaire de créer une feuille de style spéciale pour les anciennes versions d'Internet Explorer. Il faut utiliser un CSS un peu différent pour que les vieilles versions d'IE « comprennent » ce qu'il faut faire.

En créant une feuille de style spéciale pour les vieilles versions d'IE (qu'on pourrait appeler `style_ie.css`) et en utilisant la technique ci-dessous, on peut reproduire le comportement des `inline-block` :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" />
        <!--[if lte IE 7]>
        <link rel="stylesheet" href="style_ie.css" />
        <![endif]-->

        <title>Zozor - Le Site Web</title>
    </head>
```

La feuille de style `style_ie.css` contiendra des déclarations comme celle-ci :

```
element
{
    display: inline;
    zoom: 1;
}
```

Cette technique doit être appliquée sur chaque élément positionné en `inline-block`.



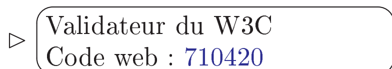
D'autres différences existent sur les vieilles versions d'IE : le texte n'est pas toujours à la bonne taille, certains blocs sont eux aussi mal dimensionnés, etc. Ces différences doivent être réglées au cas par cas dans la feuille `style_ie.css`.

Vérifier la validité

Le W3C propose sur son site web un outil appelé le « Validateur » (« Validator » en anglais). Le validateur est une sorte de programme qui va analyser votre code source et vous dire s'il est correctement écrit ou s'il comporte des erreurs que vous devez corriger.

Souvenez-vous : le W3C a établi des normes. Il est nécessaire de les respecter, pour qu'on soit sûr que tous les sites web parlent la même « langue ».

Il existe un validateur pour HTML et un validateur pour CSS (à mettre dans vos favoris !). Celui pour CSS comportant quelques bugs (il signale comme invalides des feuilles CSS qui sont tout à fait valides), nous ne nous y attarderons pas. Par contre, le validateur HTML va être très intéressant pour nous.



Vous pouvez valider votre page web de trois façons différentes, c'est pour cela qu'il y a trois onglets :

- adresse (URL) ;
- envoi du fichier `.html` ;
- copier-coller du code HTML.

Pour le moment, notre site web n'est pas encore disponible sur le Web, ce qui fait qu'il n'a pas d'adresse URL. Le mieux est donc d'envoyer le fichier `.html` que l'on a fait ou encore de copier-coller directement le code HTML.

Si vous envoyez votre code HTML et que tout se passe bien, le validateur va vous répondre avec le message représenté à la figure 14.14.

This document was successfully checked as HTML5!

FIGURE 14.14 – Le validateur du W3C nous informe que notre page ne comporte pas d'erreur

Dans ce cas, cela signifie que tout va bien et que vous avez bien construit votre page !

Malheureusement, il arrivera souvent que vous ayez des erreurs. Dans ce cas, évitez de paniquer comme cela :



AU SEEECOUUUUUUUUURS ! ! ! Ma page web n'est pas valide, je ne vais pas m'en sortir, je suis cerné par les erreurs, faites quelque chose aidez-mmmoiiiiii !

Vous aviez une belle page web, elle s'affichait bien, elle était jolie, et pourtant le validateur vous répond avec un message rouge inquiétant, en vous disant que votre page web n'est pas bien construite.

Tout d'abord, mettez-vous bien ceci en tête : ce n'est pas parce que votre page web s'affiche correctement qu'elle ne comporte pas d'erreur. Votre page web peut être toute belle et comporter beaucoup d'erreurs.



Quel intérêt de les corriger alors ?

Il faut savoir que les navigateurs « essaient » de ne pas afficher les erreurs, lorsqu'ils en rencontrent, pour ne pas perturber l'internaute. Mais rien ne vous dit que d'autres navigateurs ne vont pas se comporter bizarrement !

Avoir une page web valide, c'est donc avoir la possibilité de dormir tranquille en sachant que l'on a bien fait les choses comme il faut. Cela simplifie le travail des programmes qui lisent les pages web. De plus, et c'est vérifié, une page web correctement construite aura plus de chances d'être mieux positionnée dans les résultats de recherche de Google, ce qui vous amènera... plus de visiteurs !

Voici une liste de conseils qui peuvent vous aider à résoudre les erreurs qui risquent de vous être signalées tôt ou tard :

– Tous vos textes doivent en général être dans des balises de paragraphes. Il est interdit de mettre du texte directement entre les balises `<body></body>` sans l'avoir entouré des fameux `<p></p>`. Ceci est aussi valable pour les retours à la ligne `<br />`, qui doivent être à l'intérieur de paragraphes. C'est une erreur ultra-courante chez les débutants.

```
<p>Ceci est un texte correctement placé dans un paragraphe.
<br />
Les balises <br /> doivent se trouver à l'intérieur d'un paragraphe, ne l'oubliez pas</p>

Ceci est un texte en-dehors d'un paragraphe. C'est interdit.
<br />
```

– *Toutes vos images doivent comporter un attribut* `alt` qui indique ce que contient l'image. Si, par hasard, votre image est purement décorative (vous ne pouvez pas en trouver de description), vous êtes autorisés à ne rien mettre comme valeur pour l'attribut `alt`.

```
<!-- L'image comporte une description -->
<img src="photo.jpg" alt="Une photo de moi" />

<!-- L'image ne comporte pas de description mais a quand même un attribut alt -->
<img src="deco.png" alt="" />
```

– Vos balises doivent être *fermées dans l'ordre.*

```
<!-- Les balises ne sont pas fermées dans leur ordre d'ouverture -->
<p>Texte <em>important</p></em>

<!-- Les balises sont fermées dans leur ordre d'ouverture -->
<p>Texte <em>important</em></p>
```

Gardez bien ce schéma en tête, beaucoup de débutants font cette erreur.

– Si vos liens comportent des `&`, vous devez les remplacer par le code `&` pour éviter toute confusion au navigateur.

```
<!-- Exemple d'un mauvais lien en HTML -->
<a href="http://www.site.com/?jour=15&mois=10&an=2000">

<!-- Exemple d'un bon lien en HTML -->
<a href="http://www.site.com/?jour=15&amp;mois=10&amp;an=2000
    ">
```

– Vérifiez enfin que vous n'avez pas utilisé des balises anciennes et désormais obsolètes en HTML5 (comme le vieux `<frame>`, la balise `<marquee>`... Le validateur vous dira « Element *XXX* undefined » (balise inconnue) ou encore « There is no attribute *XXX* » (attribut inconnu).

Tout le monde fait des erreurs, alors ne paniquez pas. Corrigez pas à pas votre page web jusqu'à ce que le validateur vous affiche un beau résultat en vert.

Le code final

Je mets à disposition le code final de la page web que nous avons réalisée. Vous pouvez également visualiser le résultat (figure 14.15) en ligne grâce à un code web.



FIGURE 14.15 – Apparence finale du site web

▷ Accéder au site en ligne
Code web : 708495

Vous pouvez aussi télécharger un fichier ZIP contenant tous les fichiers du site pour pouvoir le tester chez vous.

▷ Télécharger le ZIP
Code web : 975009



Pour ajuster au mieux le site sur les anciennes versions d'Internet Explorer, vous verrez je n'ai pas créé dans ce code final de feuille `style_ie.css`. À la place, j'ai utilisé une autre technique qui consiste à donner une classe spéciale (comme `.ie7`) à la balise `<body>` uniquement sur les anciennes versions d'IE, ce qui me permet de savoir quelle est la version du navigateur dans le fichier CSS (`.ie7 footer` permet par exemple de modifier le style du footer sur IE7).

Quatrième partie

Fonctionnalités évoluées

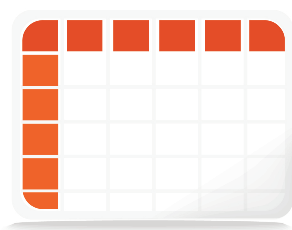
Chapitre 15

Les tableaux

Difficulté :

Indispensables pour organiser les informations, les tableaux sont un petit peu délicats à construire en HTML : cela explique que je vous les présente seulement maintenant. Il va en effet falloir imbriquer de nouvelles balises HTML dans un ordre précis.

Nous allons commencer par construire des tableaux basiques, puis nous les complexifierons au fur et à mesure : fusion de cellules, division en multiples sections... Nous découvrirons aussi les propriétés CSS liées aux tableaux, qui nous permettront de personnaliser leur apparence.



Un tableau simple

La première balise à connaître est `<table> </table>`. C'est cette balise qui permet d'indiquer le début et la fin d'un tableau.

Cette balise est de type bloc, il faut donc la placer en dehors d'un paragraphe. Exemple :

```
<p>Ceci est un paragraphe avant le tableau.</p>

<table>
   <!-- Ici, on écrira le contenu du tableau -->
</table>

<p>Ceci est un paragraphe après le tableau.</p>
```



Bon, et qu'écrit-on à l'intérieur du tableau ?

Là, préparez-vous à subir une avalanche de nouvelles balises. Pour commencer en douceur, voici deux nouvelles balises très importantes :

- `<tr> </tr>` : indique le début et la fin d'une ligne du tableau ;
- `<td> </td>` : indique le début et la fin du contenu d'une cellule.

En HTML, un tableau se construit ligne par ligne. Dans chaque ligne (`<tr>`), on indique le contenu des différentes cellules (`<td>`). Schématiquement, un tableau se construit comme à la figure 15.1.

Td (cellules)

Tr (ligne)

Nom	Age	Pays
Anne	27 ans	France
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis
Ogasaku Nyagatosoka	18 ans	Japon

Figure 15.1 – Un tableau, avec des cellules contenues dans des lignes

On a une balise de ligne (`<tr>`) qui englobe un groupe de cellules (`<td>`). Par exemple, si je veux faire un tableau à deux lignes, avec trois cellules par ligne (donc trois colonnes), je devrai taper ceci :

```
<table>
   <tr>
       <td>Carmen</td>
       <td>33 ans</td>
       <td>Espagne</td>
```

```
    </tr>
    <tr>
        <td>Michelle</td>
        <td>26 ans</td>
        <td>États-Unis</td>
    </tr>
</table>
```

Le résultat est un peu déprimant (figure 15.2).

Carmen 33 ans Espagne
Michelle 26 ans Etats-Unis

Figure 15.2 – Un tableau sans bordures



C'est un tableau ça ? Le texte s'est écrit à la suite et il n'y a même pas de bordures !

Oui, un tableau sans CSS paraît bien vide. Et justement, rajouter des bordures est très simple, vous connaissez déjà le code CSS correspondant !

```
td /* Toutes les cellules des tableaux... */
{
    border: 1px solid black; /* auront une bordure de 1px */
}
```

Et voici le résultat à la figure 15.3.

Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

Figure 15.3 – Chaque cellule a sa propre bordure

Hum, ce n'est pas encore aussi parfait que ce qu'on voudrait. En effet, on aimerait qu'il n'y ait qu'une seule bordure entre deux cellules, or ce n'est pas le cas ici.

Heureusement, il existe une propriété CSS spécifique aux tableaux, `border-collapse`, qui signifie « coller les bordures entre elles ». Cette propriété peut prendre deux valeurs :

- `collapse` : les bordures seront collées entre elles, c'est l'effet qu'on recherche ici ;
- `separate` : les bordures seront dissociées (valeur par défaut)

```
table
{
    border-collapse: collapse; /* Les bordures du tableau
        seront collées (plus joli) */
}
```

```
td
{
    border: 1px solid black;
}
```

La figure 15.4 représente le résultat obtenu.

Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

FIGURE 15.4 – Les bordures sont collées les unes aux autres

Voilà qui est mieux !

La ligne d'en-tête

Maintenant que l'on a ce qu'on voulait, on va rajouter la ligne d'en-tête du tableau. Dans l'exemple ci-dessous, les en-têtes sont « Nom », « Âge » et « Pays ». La ligne d'en-tête se crée avec un `<tr>` comme on l'a fait jusqu'ici, mais les cellules qu'elle contient sont, cette fois, encadrées par des balises `<th>` et non pas `<td>` !

```
<table>
   <tr>
       <th>Nom</th>
       <th>Âge</th>
       <th>Pays</th>
   </tr>

   <tr>
       <td>Carmen</td>
       <td>33 ans</td>
       <td>Espagne</td>
   </tr>
   <tr>
       <td>Michelle</td>
       <td>26 ans</td>
       <td>États-Unis</td>
   </tr>
</table>
```

La ligne d'en-tête est très facile à reconnaître pour deux raisons :

- les cellules sont des `<th>` au lieu des `<td>` habituels ;
- c'est la première ligne du tableau (c'est idiot, mais encore faut-il le préciser).

Comme le nom des cellules est un peu différent pour l'en-tête, il faut penser à mettre à jour le CSS pour lui dire d'appliquer une bordure sur les cellules normales *et* sur l'en-tête (figure 15.5).

```
table
```

```
{
    border-collapse: collapse;
}
td, th /* Mettre une bordure sur les td ET les th */
{
    border: 1px solid black;
}
```

Nom	Age	Pays
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

FIGURE 15.5 – Un tableau avec un en-tête

Comme vous pouvez le constater, votre navigateur a mis en gras le texte des cellules d'en-tête. C'est ce que font en général les navigateurs mais, si vous le désirez, vous pouvez changer cela à coups de CSS : modifier la couleur de fond des cellules d'en-tête, leur police, leur bordure, etc.

Titre du tableau

Normalement, tout tableau doit avoir un titre. Le titre permet de renseigner rapidement le visiteur sur le contenu du tableau. Notre exemple est constitué d'une liste de personnes... oui mais alors ? Qu'est-ce que cela représente ? Sans titre de tableau, vous le voyez, on est un peu perdu.

Heureusement, il y a <caption> ! Cette balise se place tout au début du tableau, juste avant l'en-tête. C'est elle qui contient le titre du tableau (figure 15.6) :

```
<table>
   <caption>Passagers du vol 377</caption>

   <tr>
       <th>Nom</th>
       <th>Âge</th>
       <th>Pays</th>
   </tr>
   <tr>
       <td>Carmen</td>
       <td>33 ans</td>
       <td>Espagne</td>
   </tr>
   <tr>
       <td>Michelle</td>
       <td>26 ans</td>
       <td>États-Unis</td>
   </tr>
</table>
```

Passagers du vol 377		
Nom	**Age**	**Pays**
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis

FIGURE 15.6 – Un tableau avec un titre

C'est quand même plus clair !

Sachez que vous pouvez changer la position du titre avec la propriété CSS `caption-side` qui peut prendre quatre valeurs :

- `top` : le titre sera placé au-dessus du tableau (par défaut) ;
- `bottom` : le titre sera placé en dessous du tableau ;
- `left` : le titre sera placé à gauche du tableau ;
- `right` : le titre sera placé à droite du tableau.

Un tableau structuré

Nous avons appris à construire des petits tableaux simples. Ces petits tableaux suffisent dans la plupart des cas, mais il arrivera que vous ayez besoin de réaliser des tableaux plus. . . complexes.

Nous allons découvrir deux techniques particulières :

- Pour les gros tableaux, il est possible de les **diviser** en trois parties :
  - En-tête ;
  - Corps du tableau ;
  - Pied de tableau.
- Pour certains tableaux, il se peut que vous ayez besoin de **fusionner** des cellules entre elles.

Diviser un gros tableau

Si votre tableau est assez gros, vous aurez tout intérêt à le découper en plusieurs parties. Pour cela, il existe des balises HTML qui permettent de définir les trois « zones » du tableau :

- **l'en-tête (en haut)** : il se définit avec les balises `<thead></thead>` ;
- **le corps (au centre)** : il se définit avec les balises `<tbody></tbody>` ;
- **le pied du tableau (en bas)** : il se définit avec les balises `<tfoot></tfoot>`.

Que mettre dans le pied de tableau ? Généralement, si c'est un long tableau, vous y recopiez les cellules d'en-tête. Cela permet de voir, même en bas du tableau, à quoi se rapporte chacune des colonnes. Schématiquement, un tableau en trois parties se découpe donc comme illustré à la figure 15.7.

C'est un peu déroutant mais il est conseillé d'écrire les balises dans l'ordre suivant :

Passagers du vol 377

Nom	Age	Pays
Carmen	33 ans	Espagne
Michelle	26 ans	Etats-Unis
François	43 ans	France
Martine	34 ans	France
Jonathan	13 ans	Australie
Xu	19 ans	Chine
Nom	Age	Pays

En-tête du tableau — <thead>
Corps du tableau — <tbody>
Pied du tableau — <tfoot>

FIGURE 15.7 – Un tableau découpé en plusieurs parties

1. <thead>
2. <tfoot>
3. <tbody>

Dans le code, on renseigne donc d'abord la partie du haut, ensuite la partie du bas, et enfin la partie principale (<tbody>). Le navigateur se chargera d'afficher chaque élément au bon endroit, ne vous inquiétez pas.

Voici donc le code à écrire pour construire le tableau en trois parties :

```
<table>
   <caption>Passagers du vol 377</caption>

   <thead> <!-- En-tête du tableau -->
       <tr>
           <th>Nom</th>
           <th>Âge</th>
           <th>Pays</th>
       </tr>
   </thead>

   <tfoot> <!-- Pied de tableau -->
       <tr>
           <th>Nom</th>
           <th>Âge</th>
           <th>Pays</th>
       </tr>
   </tfoot>

   <tbody> <!-- Corps du tableau -->
       <tr>
           <td>Carmen</td>
           <td>33 ans</td>
           <td>Espagne</td>
       </tr>
       <tr>
           <td>Michelle</td>
```

```
                <td>26 ans</td>
                <td>États-Unis</td>
            </tr>
            <tr>
                <td>François</td>
                <td>43 ans</td>
                <td>France</td>
            </tr>
            <tr>
                <td>Martine</td>
                <td>34 ans</td>
                <td>France</td>
            </tr>
            <tr>
                <td>Jonathan</td>
                <td>13 ans</td>
                <td>Australie</td>
            </tr>
            <tr>
                <td>Xu</td>
                <td>19 ans</td>
                <td>Chine</td>
            </tr>
        </tbody>
    </table>
```



Il n'est pas obligatoire d'utiliser ces trois balises (<thead>, <tbody>, <tfoot>) dans tous les tableaux. En fait, vous vous en servirez surtout si votre tableau est assez gros et que vous avez besoin de l'organiser plus clairement. Pour les « petits » tableaux, vous pouvez garder sans problème l'organisation plus simple que nous avons vue au début.

3, 2, 1... Fusioooon !

Dans certains tableaux complexes, vous aurez besoin de « fusionner » des cellules entre elles. Un exemple de fusion ? Regardez le tableau à la figure 15.8, qui dresse une liste de films et indique à qui ils s'adressent.

Titre du film	Pour enfants ?	Pour adolescents ?
Massacre à la tronçonneuse	Non, trop violent	Oui
Les bisounours font du ski	Oui, adapté	Pas assez violent...
Lucky Luke, seul contre tous	Pour toute la famille !	

FIGURE 15.8 – Un tableau contenant des titres de films et leur public

Pour le dernier film, vous voyez que les cellules ont été fusionnées : elles ne font plus qu'une. C'est exactement l'effet qu'on cherche à obtenir.

Pour effectuer une fusion, on rajoute un attribut à la balise `<td>`. Il faut savoir qu'il existe deux types de fusion :

- **La fusion de colonnes** : c'est ce que je viens de faire dans cet exemple. La fusion s'effectue horizontalement. On utilisera l'attribut `colspan`.
- **La fusion de lignes** : là, deux lignes seront groupées entre elles. La fusion s'effectuera verticalement. On utilisera l'attribut `rowspan`.

Comme vous le savez, vous devez donner une valeur à l'attribut (que ce soit `colspan` ou `rowspan`). Il faut indiquer le nombre de cellules à fusionner entre elles. Dans notre exemple, nous avons fusionné deux cellules : celle de la colonne « Pour enfants ? » et celle de « Pour adolescents ? ». On devra donc écrire :

```
<td colspan="2">
```

... qui signifie : « Cette cellule est la fusion de deux cellules ». Il est possible de fusionner plus de cellules à la fois (trois, quatre, cinq... autant que vous voulez).

Voilà le code HTML qui me permet de réaliser la fusion correspondant au tableau précédent :

```
1  <table>
2      <tr>
3          <th>Titre du film</th>
4          <th>Pour enfants ?</th>
5          <th>Pour adolescents ?</th>
6      </tr>
7      <tr>
8          <td>Massacre à la tronçonneuse</td>
9          <td >Non, trop violent</td>
10         <td>Oui</td>
11     </tr>
12     <tr>
13         <td>Les bisounours font du ski</td>
14         <td>Oui, adapté</td>
15         <td>Pas assez violent...</td>
16     </tr>
17     <tr>
18         <td>Lucky Luke, seul contre tous</td>
19         <td colspan="2">Pour toute la famille !</td>
20     </tr>
21 </table>
```

Une remarque importante : vous voyez que la ligne 19 ne contient que deux cellules au lieu de trois (il n'y a que deux balises `<td>`). C'est tout à fait normal car j'ai fusionné les deux dernières cellules entre elles. Le `<td colspan="2">` indique que cette cellule prend la place de deux cellules à la fois.



Et pour la fusion verticale avec `rowspan`, on fait comment ?

Cela se complique un petit peu. Pour notre exemple, nous allons « inverser » l'ordre de notre tableau : au lieu de mettre les titres de films à gauche, on va les placer en haut. C'est une autre façon de voir le tableau : au lieu de le construire en hauteur, on peut le construire en longueur.

Dans ce cas, le `colspan` n'est plus adapté, c'est un `rowspan` qu'il faut utiliser :

```
<table>
    <tr>
        <th>Titre du film</th>
        <td>Massacre à la tronçonneuse</td>
        <td>Les bisounours font du ski</td>
        <td>Lucky Luke, seul contre tous</td>
    </tr>
    <tr>
        <th>Pour enfants ?</th>
        <td>Non, trop violent</td>
        <td>Oui, adapté</td>
        <td rowspan="2">Pour toute la famille !</td>
    </tr>
    <tr>
        <th>Pour adolescents ?</th>
        <td>Oui</td>
        <td>Pas assez violent...</td>
    </tr>
</table>
```

Résultat : les cellules sont fusionnées verticalement (figure 15.9) !

<table>
<tr><th>Titre du film</th><td>Massacre à la tronçonneuse</td><td>Les bisounours font du ski</td><td>Lucky Luke, seul contre tous</td></tr>
<tr><th>Pour enfants ?</th><td>Non, trop violent</td><td>Oui, adapté</td><td rowspan="2">Pour toute la famille !</td></tr>
<tr><th>Pour adolescents ?</th><td>Oui</td><td>Pas assez violent...</td></tr>
</table>

FIGURE 15.9 – Les cellules ont été fusionnées verticalement



Notez qu'on peut modifier l'alignement vertical du texte des cellules de tableaux avec la propriété `vertical-align` que nous avons découverte dans le chapitre sur la mise en page.

En résumé

- Un tableau s'insère avec la balise `<table>` et se définit ligne par ligne avec `<tr>`.
- Chaque ligne comporte des cellules `<td>` (cellules normales) ou `<th>` (cellules d'en-tête).
- Le titre du tableau se définit avec `<caption>`.
- On peut ajouter une bordure aux cellules du tableau avec `border`. Pour fusionner les bordures, on utilise la propriété CSS `border-collapse`.

- Un tableau peut être divisé en trois sections : `<thead>` (en-tête), `<tbody>` (corps) et `<tfoot>` (bas du tableau). L'utilisation de ces balises n'est pas obligatoire.
- On peut fusionner des cellules horizontalement avec l'attribut `colspan` ou verticalement avec `rowspan`. Il faut indiquer combien de cellules doivent être fusionnées.

Chapitre 16

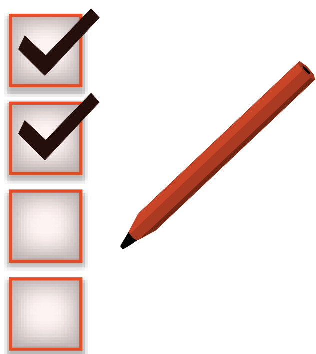
Les formulaires

Difficulté :

Toute page HTML peut être enrichie de formulaires interactifs, qui invitent vos visiteurs à renseigner des informations : saisir du texte, sélectionner des options, valider avec un bouton. . . tout est possible !

Nous arrivons cependant aux limites du langage HTML car il faut ensuite pouvoir analyser les informations que le visiteur a saisies. . . et cela ne peut pas se faire en langage HTML. Comme nous allons le voir, le traitement des résultats doit s'effectuer dans un autre langage, par exemple le PHP.

En attendant, nous avons un grand nombre de nouvelles balises HTML à découvrir. Bienvenue dans le monde merveilleux des formulaires, un monde où les boutons, les cases à cocher et les listes déroulantes vivent en harmonie (enfin presque).



Créer un formulaire

Lorsqu'il vous prend subitement l'envie d'insérer un formulaire dans votre page HTML, vous devez pour commencer écrire une balise `<form> </form>`. C'est la balise principale du formulaire, elle permet d'en indiquer le début et la fin.

```
<p>Texte avant le formulaire</p>

<form>
   <p>Texte à l'intérieur du formulaire</p>
</form>

<p>Texte après le formulaire</p>
```



Notez qu'il faut obligatoirement mettre des balises de type block (comme <p> </p>) à l'intérieur de votre formulaire si vous souhaitez y faire figurer du texte.

Voilà pour la structure de base. Maintenant, soyez attentifs : ce que j'ai à vous dire n'est pas évident parce qu'on est à la limite du HTML.

On va prendre un exemple pour que les choses soient claires. Supposons que votre visiteur vienne de taper un commentaire dans votre formulaire, par exemple un message qu'il aimerait publier sur vos forums. Ce message doit être *envoyé* pour que vous puissiez le recevoir (logique, non ?) et l'afficher pour vos autres visiteurs.

Eh bien c'est là le problème, ou plutôt les problèmes, que l'on va se poser :

- **Problème nº 1** : comment envoyer le texte saisi par le visiteur ? Par quel moyen ?
- **Problème nº 2** : une fois que les données ont été envoyées, comment les traiter ? Souhaitez-vous recevoir le message automatiquement par mail ou préférez-vous qu'un programme se charge de l'enregistrer quelque part, puis de l'afficher sur une page visible par tout le monde ?

Pour fournir les réponses à ces deux problèmes, vous devez ajouter deux attributs à la balise `<form>` :

- `method` : cet attribut indique par quel moyen les données vont être envoyées (réponse au **problème nº 1**). Il existe deux solutions pour envoyer des données sur le Web :
  - `method="get"` : c'est une méthode en général assez peu adaptée car elle est limitée à 255 caractères. La particularité vient du fait que les informations seront envoyées dans l'adresse de la page (`http://...`), mais ce détail ne nous intéresse pas vraiment pour le moment. La plupart du temps, je vous recommande d'utiliser l'autre méthode : `post`.
  - `method="post"` : c'est la méthode la plus utilisée pour les formulaires car elle permet d'envoyer un grand nombre d'informations. Les données saisies dans le formulaire ne transitent pas par la barre d'adresse.
- `action` : c'est l'adresse de la page ou du programme qui va *traiter* les informations (réponse au **problème nº 2**). Cette page se chargera de vous envoyer un e-mail avec le message si c'est ce que vous voulez, ou bien d'enregistrer le message avec tous les

autres dans une base de données. Cela ne peut pas se faire en HTML et CSS, on utilisera en général un autre langage dont vous avez peut-être entendu parler : PHP.

On va donc maintenant compléter la balise `<form>` avec les deux attributs qu'on vient de voir.

Pour `method`, vous l'aurez deviné, je vais mettre la valeur `post`. Pour `action`, je vais taper le nom d'une page fictive en PHP (`traitement.php`). C'est cette page qui sera appelée lorsque le visiteur cliquera sur le bouton d'envoi du formulaire.

```
<p>Texte avant le formulaire</p>

<form method="post" action="traitement.php">
   <p>Texte à l'intérieur du formulaire</p>
</form>

<p>Texte après le formulaire</p>
```

Pour le moment, on ne sait pas ce qu'il se passe à l'intérieur de la page `traitement.php` : je vous demande de me faire confiance et d'imaginer que cette page existe et fonctionne.

Notre priorité, pour le moment, est de découvrir en HTML/CSS comment faire pour insérer des zones de texte, des boutons et des cases à cocher dans votre page web. C'est ce que nous allons voir maintenant.

Les zones de saisie basiques

Bien, retour au concret.

Nous allons passer en revue les différentes balises HTML permettant de saisir du texte dans un formulaire. Il faut savoir qu'il y a deux zones de texte différentes :

- **La zone de texte monoligne** : comme son nom l'indique, on ne peut y écrire qu'une seule ligne. Elle sert à saisir des textes courts, par exemple un pseudo.
- **La zone de texte multiligne** : cette zone de texte permet d'écrire une quantité importante de texte sur plusieurs lignes, par exemple une dissertation sur l'utilité du HTML dans le développement des pays d'Asie du Sud-Est (ce n'est qu'une suggestion hein...).

Zone de texte monoligne

La figure 16.1 montre à quoi ressemble une zone de texte monoligne.

Votre pseudo :

FIGURE 16.1 – Une zone de texte monoligne

Pour insérer une zone de texte dans une ligne, on va utiliser la balise `<input />`.



On retrouvera cette balise plusieurs fois dans la suite de ce chapitre. À chaque fois, c'est la valeur de son attribut type qui va changer.

Pour créer une zone de texte à une ligne, on doit écrire :

```
<input type="text" />
```

Ce n'est pas encore suffisant : il faut donner un nom à votre zone de texte. Ce nom n'apparaît pas sur la page mais il vous sera indispensable par la suite. En effet, cela vous permettra (en PHP par exemple) de reconnaître d'où viennent les informations : vous saurez que tel texte est le pseudo du visiteur, tel texte est le mot de passe qu'il a choisi, etc.

Pour donner un nom à un élément de formulaire, on utilise l'attribut name. Ici, on va supposer qu'on demande au visiteur de rentrer son pseudo :

```
<input type="text" name="pseudo" />
```

Essayons donc de créer un formulaire très basique avec notre champ de texte :

```
<form method="post" action="traitement.php">
   <p><input type="text" name="pseudo" /></p>
</form>
```

Comme d'habitude, je vous invite fortement à essayer ce code chez vous afin d'observer le résultat.

Les libellés

Cette zone de texte est bien jolie mais si votre visiteur tombe dessus, il ne sait pas ce qu'il doit écrire. C'est justement le rôle de la balise <label> :

```
<form method="post" action="traitement.php">
    <p>
        <label>Votre pseudo</label> : <input type="text" name="pseudo" />
    </p>
</form>
```

Ce code donne exactement le résultat que vous avez pu observer à la figure 16.1.

Mais cela ne suffit pas. Il faut lier le label à la zone de texte. Pour ce faire, on doit donner un nom à la zone de texte, non pas avec l'attribut name mais avec l'attribut id (que l'on peut utiliser sur toutes les balises).



Un name et un id sur le champ ? Cela ne va-t-il pas faire double emploi ?

Si, un peu. Personnellement, je donne la même valeur au `name` et à l'`id`, cela ne pose pas de problème.

Pour lier le label au champ, il faut lui donner un attribut `for` qui a la même valeur que l'`id` du champ... Le mieux est de le voir sur un exemple :

```
<form method="post" action="traitement.php">
   <p>
       <label for="pseudo">Votre pseudo</label> : <input type="text" name="pseudo" id="pseudo" />
   </p>
</form>
```

Essayez de cliquer sur le texte « Votre pseudo » : vous allez voir que le curseur se place automatiquement dans la zone de texte correspondante.

Quelques attributs supplémentaires

On peut ajouter un certain nombre d'autres attributs à la balise `<input />` pour personnaliser son fonctionnement :

- On peut agrandir le champ avec `size`.
- On peut limiter le nombre de caractères que l'on peut saisir avec `maxlength`.
- On peut pré-remplir le champ avec une valeur par défaut à l'aide de `value`.
- On peut donner une indication sur le contenu du champ avec `placeholder`. Cette indication disparaîtra dès que le visiteur aura cliqué à l'intérieur du champ.

Dans l'exemple suivant, la zone de texte contient une indication permettant de comprendre ce qu'il faut saisir ; le champ fait 30 caractères de long mais on ne peut écrire que 10 caractères maximum à l'intérieur :

```
<form method="post" action="traitement.php">
    <p>
        <label for="pseudo">Votre pseudo :</label>
        <input type="text" name="pseudo" id="pseudo" placeholder="Ex : Zozor" size="30" maxlength="10" />
    </p>
</form>
```

Testez vous-mêmes le résultat pour observer le comportement du champ. En attendant, voici le rendu du champ dans son état initial en figure 16.2.

Votre pseudo : Ex : Zozor

FIGURE 16.2 – Un champ de texte avec une indication `placeholder`

Zone de mot de passe

Vous pouvez facilement faire en sorte que la zone de texte se comporte comme une « zone de mot de passe », c'est-à-dire une zone où on ne voit pas à l'écran les caractères saisis. Pour créer ce type de zone de saisie, utilisez l'attribut `type="password"`.

Je complète mon formulaire. Il demande maintenant au visiteur son pseudo *et* son mot de passe :

```
<form method="post" action="traitement.php">
   <p>
       <label for="pseudo">Votre pseudo :</label>
       <input type="text" name="pseudo" id="pseudo" />

       <br />
       <label for="pass">Votre mot de passe :</label>
       <input type="password" name="pass" id="pass" />

   </p>
</form>
```

Testez la zone de mot de passe : vous verrez que les caractères ne s'affichent pas à l'écran, comme sur la figure 16.3.

Votre pseudo : M@teo21
Votre mot de passe : •••••••••••••••

FIGURE 16.3 – Une zone de saisie de mot de passe a été ajoutée

Zone de texte multiligne

Pour créer une zone de texte multiligne, on change de balise : nous allons utiliser `<textarea> </textarea>`.

Comme pour tout autre élément du formulaire, il faut lui donner un nom avec `name` et utiliser un `label` qui explique de quoi il s'agit.

```
<form method="post" action="traitement.php">
   <p>
       <label for="ameliorer">Comment pensez-vous que je
           pourrais améliorer mon site ?</label><br />
       <textarea name="ameliorer" id="ameliorer"></textarea>
   </p>
</form>
```

Et voici le résultat en image (figure 16.4) !

Comme vous pouvez le constater, c'est un peu petit ! Heureusement, on peut modifier la taille du `<textarea>` de deux façons différentes.

Comment pensez-vous que je pourrais améliorer mon site ?

FIGURE 16.4 – Une (petite) zone de saisie multiligne

- **En CSS** : il suffit d'appliquer les propriétés CSS `width` et `height` au `<textarea>`.
- **Avec des attributs** : on peut ajouter les attributs `rows` et `cols` à la balise `<textarea>`. Le premier indique le nombre de lignes de texte qui peuvent être affichées simultanément, et le second le nombre de colonnes.



Pourquoi ouvre-t-on la balise `<textarea>` pour la fermer juste après ?

Vous pouvez pré-remplir le `<textarea>` avec une valeur par défaut. Dans ce cas, on n'utilise pas l'attribut `value` : on écrit tout simplement le texte par défaut entre la balise ouvrante et la balise fermante !

```
<form method="post" action="traitement.php">
   <p>
       <label for="ameliorer">
       Comment pensez-vous que je puisse améliorer mon site ?
       </label>
       <br />

       <textarea name="ameliorer" id="ameliorer" rows="10" cols="50">
       Améliorer ton site ?!
       Mais enfin ! Il est tellement génialissime qu'il n'est pas nécessaire de l'améliorer !
       </textarea>
   </p>
</form>
```

Et voici le résultat à la figure 16.5.

Comment pensez-vous que je puisse améliorer mon site ?

Améliorer ton site ?!
Mais enfin ! Il est tellement génialissime qu'il
n'est pas nécessaire de l'améliorer !

FIGURE 16.5 – Une zone de saisie multiligne pré-remplie

Les zones de saisie enrichies

HTML5 apporte de nombreuses fonctionnalités nouvelles relatives aux formulaires. De nouveaux types de champs sont en effet apparus avec cette version. Il suffit de donner à l'attribut `type` de la balise `<input />` l'une des nouvelles valeurs disponibles. Faisons un petit tour d'horizon !



Tous les navigateurs ne connaissent pas encore ces zones de saisie enrichies. À leur place, les anciennes versions des navigateurs afficheront une simple zone de saisie monoligne (comme si on avait écrit `type="text"`). Entre nous, c'est parfait : les nouveaux navigateurs peuvent profiter des dernières fonctionnalités, tandis que les anciens affichent une zone de texte de remplacement qui convient tout aussi bien. Vous avez donc tout intérêt à utiliser ces nouvelles zones de saisie dès aujourd'hui ! Au mieux, vos visiteurs profiteront des nouvelles fonctionnalités, au pire, ils ne verront aucun problème.

E-mail

Vous pouvez demander à saisir une adresse e-mail :

```
<input type="email" />
```

Le champ vous semblera *a priori* identique mais votre navigateur sait désormais que l'utilisateur doit saisir une adresse e-mail. Il peut afficher une indication si l'adresse n'est pas un e-mail, c'est ce que fait Firefox par exemple (figure 16.6).

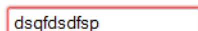


FIGURE 16.6 – Un champ e-mail mal renseigné est entouré de rouge dans Firefox

Sachez que certains navigateurs, comme les navigateurs mobiles sur iPhone et Android, affichent un clavier adapté à la saisie d'e-mail (figure 16.7).

Une URL

Avec le type `url`, on peut demander à saisir une adresse absolue (commençant généralement par `http://`) :

```
<input type="url" />
```

Même principe : si le champ ne vous semble pas différent sur votre ordinateur, sachez que celui-ci comprend bel et bien que le visiteur est censé saisir une adresse. Les navigateurs mobiles affichent par exemple un clavier adapté à la saisie d'URL (figure 16.8).



FIGURE 16.7 – Clavier de saisie d'e-mail sur un iPhone



FIGURE 16.8 – Clavier de saisie d'URL sur iPhone

Numéro de téléphone

Ce champ est dédié à la saisie de numéros de téléphone :

```
<input type="tel" />
```

Sur iPhone, par exemple, un clavier adapté s'affiche lorsqu'on doit remplir le champ (figure 16.9).



FIGURE 16.9 – Clavier de saisie de numéro de téléphone sur un iPhone

Nombre

Ce champ permet de saisir un nombre entier :

```
<input type="number" />
```

Le champ s'affichera en général avec des petites flèches pour changer la valeur (figure 16.10).



FIGURE 16.10 – Champ de saisie de nombre

Vous pouvez personnaliser le fonctionnement du champ avec les attributs suivants :

- `min` : valeur minimale autorisée.
- `max` : valeur maximale autorisée.
- `step` : c'est le « pas » de déplacement. Si vous indiquez un pas de 2, le champ n'acceptera que des valeurs de 2 en 2 (par exemple 0, 2, 4, 6...).

Un curseur

Le type `range` permet de sélectionner un nombre avec un curseur (aussi appelé *slider*), comme à la figure 16.11 :

```
<input type="range" />
```



Figure 16.11 – Un curseur grâce au type range

Vous pouvez utiliser là aussi les attributs `min`, `max` et `step` pour restreindre les valeurs disponibles.

Couleur

Ce champ permet de saisir une couleur :

```
<input type="color" />
```

En pratique, il reste assez peu mis en œuvre par les navigateurs à l'heure actuelle. Ne vous étonnez pas si vous voyez seulement un champ de texte classique.

Date

Différents types de champs de sélection de date existent :

- `date` : pour la date (05/08/1985 par exemple) ;
- `time` : pour l'heure (13:37 par exemple) ;
- `week` : pour la semaine ;
- `month` : pour le mois ;
- `datetime` : pour la date et l'heure (avec gestion du décalage horaire) ;
- `datetime-local` pour la date et l'heure (sans gestion du décalage horaire).

Exemple :

```
<input type="date" />
```

Comme vous le voyez, il y a le choix ! À l'heure actuelle, peu de navigateurs gèrent ce type de champ à part Opera.

Recherche

On peut créer un champ de recherche comme ceci :

```
<input type="search" />
```

Le navigateur décide ensuite comment afficher le champ de recherche. Ainsi, il peut ajouter une petite loupe au champ pour signifier que c'est un champ de recherche et éventuellement mémoriser les dernières recherches effectuées par le visiteur.

Les éléments d'options

HTML vous offre une ribambelle d'éléments d'options à utiliser dans votre formulaire. Ce sont des éléments qui demandent au visiteur de faire un choix parmi une liste de possibilités. Nous allons passer en revue :

- les cases à cocher ;
- les zones d'options ;
- les listes déroulantes.

Les cases à cocher

Créer une case à cocher ? Rien de plus simple ! Nous allons réutiliser la balise `<input />`, en spécifiant cette fois le type `checkbox` :

```
<input type="checkbox" name="choix" />
```

Rajoutez un `<label>` bien placé, et le tour est joué !

```
<form method="post" action="traitement.php">
    <p>
        Cochez les aliments que vous aimez manger :<br />
        <input type="checkbox" name="frites" id="frites" /> <label for="frites">Frites</label><br />
        <input type="checkbox" name="steak" id="steak" /> <label for="steak">Steak haché</label><br />
        <input type="checkbox" name="epinards" id="epinards" /> <label for="epinards">Epinards</label><br />
        <input type="checkbox" name="huitres" id="huitres" /> <label for="huitres">Huitres</label>
    </p>
</form>
```

▷ Copier ce code
Code web : 664666

Et voici le résultat en figure 16.12.

Cochez les aliments que vous aimez manger :
Frites
Steak haché
Epinards
Huitres

FIGURE 16.12 – Des cases à cocher

N'oubliez pas de donner un nom différent à chaque case à cocher, cela vous permettra d'identifier plus tard lesquelles ont été cochées par le visiteur.

Enfin, sachez que vous pouvez faire en sorte qu'une case soit cochée par défaut avec l'attribut `checked` :

```
<input type="checkbox" name="choix" checked />
```



Normalement, tout attribut possède une valeur. Dans le cas présent, en revanche, ajouter une valeur n'est pas obligatoire : la présence de l'attribut suffit à faire comprendre à l'ordinateur que la case doit être cochée. Si cela vous perturbe, sachez que vous pouvez donner n'importe quelle valeur à l'attribut (certains webmasters écrivent parfois `checked="checked"` mais c'est un peu redondant !). Dans tous les cas, la case sera cochée.

Les zones d'options

Les zones d'options vous permettent de faire un choix (et un seul) parmi une liste de possibilités. Elles ressemblent un peu aux cases à cocher mais il y a une petite difficulté supplémentaire : elles doivent être organisées en groupes. Les options d'un même groupe possèdent le même nom (`name`), mais chaque option doit avoir une valeur (`value`) différente.

La balise à utiliser est toujours un `<input />`, avec cette fois la valeur `radio` pour l'attribut `type`.

Les choses seront plus claires sur l'exemple ci-dessous :

```
<form method="post" action="traitement.php">
   <p>
       Veuillez indiquer la tranche d'âge dans laquelle vous vous situez :<br />
       <input type="radio" name="age" value="moins15" id="moins15" /> <label for="moins15">Moins de 15 ans</label><br />
       <input type="radio" name="age" value="medium15-25" id="medium15-25" /> <label for="medium15-25">15-25 ans</label><br />
       <input type="radio" name="age" value="medium25-40" id="medium25-40" /> <label for="medium25-40">25-40 ans</label><br />
       <input type="radio" name="age" value="plus40" id="plus40" /> <label for="plus40">Encore plus vieux que ça ?!</label>
   </p>
</form>
```

▷ Copier ce code
Code web : 574747

Ce qui nous donne la figure 16.13.

Pourquoi avoir mis le même nom pour chaque option ? Tout simplement pour que le navigateur sache de quel « groupe » le bouton fait partie. Essayez d'enlever les attributs `name`, vous verrez qu'il devient possible de sélectionner tous les éléments d'options. Or,

Veuillez indiquer la tranche d'âge dans laquelle vous vous situez :
- Moins de 15 ans
- 15-25 ans
- 25-40 ans
- Encore plus vieux que ça ?!

FIGURE 16.13 – Des boutons radio

ce n'est pas ce que l'on veut, c'est pour cela qu'on les « lie » entre eux en leur donnant un nom identique.



Vous noterez que cette fois on a choisi un `id` différent de `name`. En effet, les valeurs de `name` étant identiques, on n'aurait pas pu différencier les *id* (et vous savez bien qu'un `id` doit être unique !). Voilà donc pourquoi on a choisi de donner à l'`id` la même valeur que `value`.

Si vous avez deux zones d'options différentes, il faut donner un `name` unique à chaque groupe, comme ceci :

```
<form method="post" action="traitement.php">
    <p>
        Veuillez indiquer la tranche d'âge dans laquelle vous
            vous situez :<br />
        <input type="radio" name="age" value="moins15" id="
            moins15" /> <label for="moins15">Moins de 15 ans</
            label><br />
        <input type="radio" name="age" value="medium15-25" id="
            medium15-25" /> <label for="medium15-25">15-25 ans</
            label><br />
        <input type="radio" name="age" value="medium25-40" id="
            medium25-40" /> <label for="medium25-40">25-40 ans</
            label><br />
        <input type="radio" name="age" value="plus40" id="plus40
            " /> <label for="plus40">Encore plus vieux que ça ?!<
            /label>
    </p>
    <p>
        Sur quel continent habitez-vous ?<br />
        <input type="radio" name="continent" value="europe" id="
            europe" /> <label for="europe">Europe</label><br />
        <input type="radio" name="continent" value="afrique" id=
            "afrique" /> <label for="afrique">Afrique</label><br
            />
        <input type="radio" name="continent" value="asie" id="
            asie" /> <label for="asie">Asie</label><br />
        <input type="radio" name="continent" value="amerique" id
            ="amerique" /> <label for="amerique">Amérique</label>
            <br />
        <input type="radio" name="continent" value="australie"
```

```
            id="australie" /> <label for="australie">Australie</
            label>
    </p>
</form>
```



L'attribut checked est, là aussi, disponible pour sélectionner une valeur par défaut.

Les listes déroulantes

Les listes déroulantes sont un autre moyen élégant de faire un choix parmi plusieurs possibilités. Le fonctionnement est un peu différent. On va utiliser la balise `<select> </select>` qui indique le début et la fin de la liste déroulante. On ajoute l'attribut `name` à la balise pour donner un nom à la liste.

Puis, à l'intérieur du `<select> </select>`, nous allons placer plusieurs balises `<option> </option>` (une par choix possible). On ajoute à chacune d'elles un attribut `value` pour pouvoir identifier ce que le visiteur a choisi.

Voici un exemple d'utilisation :

```
<form method="post" action="traitement.php">
    <p>
        <label for="pays">Dans quel pays habitez-vous ?</label><
            br />
        <select name="pays" id="pays">
            <option value="france">France</option>
            <option value="espagne">Espagne</option>
            <option value="italie">Italie</option>
            <option value="royaume-uni">Royaume-Uni</option>
            <option value="canada">Canada</option>
            <option value="etats-unis">États-Unis</option>
            <option value="chine">Chine</option>
            <option value="japon">Japon</option>
        </select>
    </p>
</form>
```

▷ Copier ce code
Code web : 184462

Le résultat obtenu est représenté à la figure 16.14.

Si vous voulez qu'une option soit sélectionnée par défaut, utilisez cette fois l'attribut `selected` :

```
<option value="canada" selected>Canada</option>
```

Dans quel pays habitez-vous ?
Espagne
France
Espagne
Italie
Royaume-Uni
Canada
Etats-Unis
Chine
Japon

FIGURE 16.14 – Une liste déroulante

Vous pouvez aussi grouper vos options avec la balise `<optgroup> </optgroup>`. Dans notre exemple, pourquoi ne pas séparer les pays en fonction de leur continent ?

```
<form method="post" action="traitement.php">
   <p>
       <label for="pays">Dans quel pays habitez-vous ?</label><br />
       <select name="pays" id="pays">
           <optgroup label="Europe">
               <option value="france">France</option>
               <option value="espagne">Espagne</option>
               <option value="italie">Italie</option>
               <option value="royaume-uni">Royaume-Uni</option>
           </optgroup>
           <optgroup label="Amérique">
               <option value="canada">Canada</option>
               <option value="etats-unis">Etats-Unis</option>
           </optgroup>
           <optgroup label="Asie">
               <option value="chine">Chine</option>
               <option value="japon">Japon</option>
           </optgroup>
       </select>
   </p>
</form>
```

Le résultat obtenu est représenté à la figure 16.15.



Les groupes ne peuvent pas être sélectionnés. Ainsi, dans notre exemple, on ne peut pas choisir « Europe » : seuls les noms de pays sont disponibles pour la sélection.

Finaliser et envoyer le formulaire

Nous y sommes presque. Il ne nous reste plus qu'à agrémenter notre formulaire de quelques dernières fonctionnalités (comme la validation), puis nous pourrons ajouter

Dans quel pays habitez-vous ?
Espagne
Europe
France
Espagne
Italie
Royaume-Uni
Amérique
Canada
Etats-Unis
Asie
Chine
Japon

FIGURE 16.15 – Les options sont regroupées par continent

le bouton d'envoi du formulaire.

Regrouper les champs

Si votre formulaire grossit et comporte beaucoup de champs, il peut être utile de les regrouper au sein de plusieurs balises `<fieldset>`. Chaque `<fieldset>` peut contenir une légende avec la balise `<legend>`. Regardez cet exemple :

```
<form method="post" action="traitement.php">

   <fieldset>
       <legend>Vos coordonnées</legend> <!-- Titre du fieldset
          -->

       <label for="nom">Quel est votre nom ?</label>
       <input type="text" name="nom" id="nom" />

       <label for="prenom">Quel est votre prénom ?</label>
       <input type="text" name="prenom" id="prenom" />

       <label for="email">Quel est votre e-mail ?</label>
       <input type="email" name="email" id="email" />

   </fieldset>

   <fieldset>
       <legend>Votre souhait</legend> <!-- Titre du fieldset
          -->

       <p>
           Faites un souhait que vous voudriez voir exaucé :

           <input type="radio" name="souhait" value="riche" id=
              "riche" /> <label for="riche">Etre riche</label>
```

```
            <input type="radio" name="souhait" value="celebre" id="celebre" /> <label for="celebre">Etre célèbre</label>
            <input type="radio" name="souhait" value="intelligent" id="intelligent" /> <label for="intelligent">Etre <strong>encore</strong> plus intelligent</label>
            <input type="radio" name="souhait" value="autre" id="autre" /> <label for="autre">Autre...</label>
        </p>

        <p>
            <label for="precisions">Si "Autre", veuillez préciser :</label>
            <textarea name="precisions" id="precisions" cols="40" rows="4"></textarea>
        </p>
    </fieldset>
</form>
```

▷ Copier ce code
Code web : 954249

Le résultat obtenu est représenté à la figure 16.16.

Figure 16.16 – Les champs sont regroupés

Sélectionner automatiquement un champ

Vous pouvez placer automatiquement le curseur dans l'un des champs de votre formulaire avec l'attribut `autofocus`. Dès que le visiteur chargera la page, le curseur se placera dans ce champ.

Par exemple, pour que le curseur soit par défaut dans le champ `prenom` :

```
<input type="text" name="prenom" id="prenom" autofocus />
```

Rendre un champ obligatoire

Vous pouvez faire en sorte qu'un champ soit obligatoire en lui donnant l'attribut `required`.

```
<input type="text" name="prenom" id="prenom" required />
```

Le navigateur indiquera alors au visiteur, si le champ est vide au moment de l'envoi, qu'il doit impérativement être rempli.



Les anciens navigateurs, qui ne reconnaissent pas cet attribut, enverront le contenu du formulaire sans vérification. Pour ces navigateurs, il sera nécessaire de compléter les tests avec, par exemple, des scripts JavaScript.



On dispose de pseudo-formats en CSS pour changer le style des éléments requis (`:required`) et invalides (`:invalid`). N'oubliez pas non plus que vous disposez du pseudo-format `:focus` pour changer l'apparence d'un champ lorsque le curseur se trouve à l'intérieur.

```
:required
{
    background-color: red;
}
```

Le bouton d'envoi

Il ne nous reste plus qu'à créer le bouton d'envoi. Là encore, la balise `<input />` vient à notre secours. Elle existe en quatre versions :

- `type="submit"` : le principal bouton d'envoi de formulaire. C'est celui que vous utiliserez le plus souvent. Le visiteur sera conduit à la page indiquée dans l'attribut `action` du formulaire.
- `type="reset"` : remise à zéro du formulaire.
- `type="image"` : équivalent du bouton `submit`, présenté cette fois sous forme d'image. Rajoutez l'attribut `src` pour indiquer l'URL de l'image.
- `type="button"` : bouton générique, qui n'aura (par défaut) aucun effet. En général, ce bouton est géré en JavaScript pour exécuter des actions sur la page. Nous ne l'utiliserons pas ici.



On peut changer le texte affiché à l'intérieur des boutons avec l'attribut `value`.

Pour créer un bouton d'envoi on écrira donc par exemple :

```
<input type="submit" value="Envoyer" />
```

Ce qui nous donne la figure 16.17.

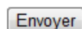


FIGURE 16.17 – Un bouton d'envoi

Lorsque vous cliquez sur le bouton « Envoyer », le formulaire vous amène alors à la page indiquée dans l'attribut `target`. Souvenez-vous, nous avions imaginé une page fictive : `traitement.php`.

Le problème, c'est que vous ne pouvez pas créer cette page seulement en HTML. Il est nécessaire d'apprendre un nouveau langage, comme le PHP, pour pouvoir « récupérer » les informations saisies et décider quoi en faire. Cela tombe bien, j'ai aussi rédigé un cours sur le langage PHP pour ceux que cela intéresse [1] !

▷ Lire le cours PHP
Code web : 955969



Certains sites comme `swisstools.net` proposent des services appelés « MailForm » qui se chargent de vous envoyer un e-mail lorsqu'un de vos visiteurs a rempli le formulaire. Cela vous dispense d'apprendre un nouveau langage mais ce n'est pas très pratique : vous devrez soit payer, soit voir de la publicité... et vous n'avez pas beaucoup de possibilités de personnaliser le traitement des données.

En résumé

- Un formulaire est une zone interactive de la page, dans laquelle vos visiteurs peuvent saisir des informations.
- On délimite un formulaire avec la balise `<form>` à laquelle il faut ajouter deux attributs : `method` (mode d'envoi des données) et `action` (page vers laquelle le visiteur sera redirigé après envoi du formulaire et qui traitera les informations).
- Une grande partie des éléments du formulaire peut s'insérer avec la balise `<input />`. La valeur de son attribut `type` permet d'indiquer quel type de champ doit être inséré :
  - `text` : zone de texte ;
  - `password` : zone de texte pour mot de passe ;
  - `tel` : numéro de téléphone ;
  - `checkbox` : case à cocher ;
  - etc.

1. Ce cours existe également en livre : `Concevez votre site web avec PHP et MySQL`, de Mathieu Nebra, paru chez Simple IT dans la collection *le Livre du Zéro*, ISBN 978-2-9535278-1-0.

- La balise `<label>` permet d'écrire un libellé. On l'associe à un champ de formulaire avec l'attribut `for`, qui doit avoir la même valeur que l'`id` du champ de formulaire.
- On peut rendre un champ obligatoire avec l'attribut `required`, faire en sorte qu'il soit sélectionné par défaut avec `autofocus`, donner une indication dans le champ avec `placeholder`...
- Pour récupérer ce que les visiteurs ont saisi, le langage HTML ne suffit pas. Il faut utiliser un langage « serveur » comme PHP... Si vous voulez aller plus loin, il va donc falloir apprendre un nouveau langage !

Chapitre 17

La vidéo et l'audio

Difficulté :

Depuis l'arrivée de Youtube et Dailymotion, il est devenu courant aujourd'hui de regarder des vidéos sur des sites web. Il faut dire que l'arrivée du haut débit a aidé à démocratiser les vidéos sur le Web.

Cependant, aucune balise HTML ne permettait jusqu'ici de gérer la vidéo. Il fallait à la place utiliser un **plugin**, comme Flash. Encore aujourd'hui, Flash reste de loin le moyen le plus utilisé pour regarder des vidéos sur Youtube, Dailymotion, Vimeo et ailleurs. Mais utiliser un plugin a de nombreux défauts : on dépend de ceux qui gèrent le plugin (en l'occurence, l'entreprise Adobe, qui possède Flash), on ne peut pas toujours contrôler son fonctionnement, il y a parfois des failles de sécurité... Au final, c'est assez lourd.

C'est pour cela que deux nouvelles balises standard ont été créées en HTML5 : `<video>` et `<audio>` !



Les formats audio et vidéo

Lorsque je vous ai présenté les images et la balise `<img />`, j'ai commencé par un petit tour d'horizon des différents formats d'images (JPEG, PNG, GIF, etc.). Pour la vidéo et l'audio, je vais faire pareil… mais c'est plus compliqué.

En fait, le fonctionnement des vidéos est même tellement complexe qu'on pourrait faire un cours entier à ce sujet ! Étant donné qu'on parle ici de HTML, nous n'allons pas passer toutes nos prochaines nuits à étudier les subtilités de l'encodage vidéo. Je vais donc simplifier les choses et vous expliquer juste ce que vous avez besoin de savoir.

Les formats audio

Pour diffuser de la musique ou n'importe quel son, il existe de nombreux formats. La plupart d'entre eux sont compressés (comme le sont les images JPEG, PNG et GIF) ce qui permet de réduire leur poids :

- **MP3** : vous ne pouvez *pas* ne pas en avoir entendu parler ! C'est l'un des plus vieux, mais aussi l'un des plus compatibles (tous les appareils savent lire des MP3), ce qui fait qu'il est toujours très utilisé aujourd'hui.
- **AAC** : utilisé majoritairement par Apple sur iTunes, c'est un format de bonne qualité. Les iPod, iPhone et autres iPad savent les lire sans problème.
- **OGG** : le format Ogg Vorbis est très répandu dans le monde du logiciel libre, notamment sous Linux. Ce format a l'avantage d'être libre, c'est-à-dire qu'il n'est protégé par aucun brevet.
- **WAV** (**format non compressé**) : évitez autant que possible de l'utiliser car le fichier est très volumineux avec ce format. C'est un peu l'équivalent du Bitmap (BMP) pour l'audio.

Aucun navigateur ne gère tous ces formats à la fois. Retenez surtout la compatibilité pour les MP3 et OGG :

Navigateur	MP3	OGG
Internet Explorer	Oui	-
Chrome	Oui	Oui
Firefox	-	Oui
Safari	Oui	-
Opera	-	Oui



Il n'y a pas de format « idéal » reconnu par tous les navigateurs ?

Eh non ! Heureusement, on pourra proposer différents formats aux navigateurs qui sélectionneront alors celui qu'ils savent lire.

Les formats vidéo

Le stockage de la vidéo est autrement plus complexe. On a besoin de trois éléments :

- **Un format conteneur** : c'est un peu comme une boîte qui va servir à contenir les deux éléments ci-dessous. On reconnaît en général le type de conteneur à l'extension du fichier : AVI, MP4, MKV...
- **Un codec audio** : c'est le format du son de la vidéo, généralement compressé. Nous venons de les voir, on utilise les mêmes : MP3, AAC, OGG...
- **Un codec vidéo** : c'est le format qui va compresser les images. C'est là que les choses se corsent, car ces formats sont complexes et on ne peut pas toujours les utiliser gratuitement. Les principaux à connaître pour le Web sont :
  - **H.264** : l'un des plus puissants et des plus utilisés aujourd'hui... mais il n'est pas 100% gratuit. En fait, on peut l'utiliser gratuitement dans certains cas (comme la diffusion de vidéos sur un site web personnel), mais il y a un flou juridique qui fait qu'il est risqué de l'utiliser à tout va.
  - **Ogg Theora** : un codec gratuit et libre de droits, mais moins puissant que H.264. Il est bien reconnu sous Linux mais, sous Windows, il faut installer des programmes pour pouvoir le lire.
  - **WebM** : un autre codec gratuit et libre de droits, plus récent. Proposé par Google, c'est le concurrent le plus sérieux de H.264 à l'heure actuelle.

Quelle est la compatibilité des codecs vidéo sur les différents navigateurs ? Là encore, vous allez voir que c'est un joyeux bazar :

Navigateur	H.264	Ogg Theora	WebM
Internet Explorer	Oui	-	Oui
Chrome	-	Oui	Oui
Firefox	-	Oui	Oui
Safari	Oui	-	-
Opera	Oui	Oui	Oui

Là encore, aucun format ne sort du lot. Il est conseillé de proposer chaque vidéo dans plusieurs formats pour qu'elle soit lisible sur un maximum de navigateurs.

Pour convertir une vidéo dans ces différents formats, je vous conseille l'excellent logiciel gratuit `Miro Video Converter` que vous pouvez télécharger grâce au code web ci-dessous.

▷ Télécharger le logiciel
Code web : 547678

Il vous suffit de glisser-déposer votre vidéo dans la fenêtre du programme et de sélectionner le format de sortie souhaité. Cela vous permettra de créer plusieurs versions de votre vidéo !

Insertion d'un élément audio



La balise <audio> que nous allons découvrir est reconnue par tous les navigateurs récents, y compris Internet Explorer à partir de la version 9 (IE9).

En théorie, il suffit d'une simple balise pour jouer un son sur notre page :

```
<audio src="musique.mp3"></audio>
```

En pratique, c'est un peu plus compliqué que cela. Si vous testez ce code... vous ne verrez rien ! En effet, le navigateur va seulement télécharger les informations générales sur le fichier (on parle de **métadonnées**) mais il ne se passera rien de particulier.

Vous pouvez compléter la balise des attributs suivants :

- `controls` : pour ajouter les boutons « Lecture », « Pause » et la barre de défilement. Cela peut sembler indispensable, et vous vous demandez peut-être pourquoi cela n'y figure pas par défaut, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript.
- `width` : pour modifier la largeur de l'outil de lecture audio.
- `loop` : la musique sera jouée en boucle.
- `autoplay` : la musique sera jouée dès le chargement de la page. Évitez d'en abuser, c'est en général irritant d'arriver sur un site qui joue de la musique tout seul !
- `preload` : indique si la musique peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
  - `auto` (par défaut) : le navigateur décide s'il doit précharger toute la musique, uniquement les métadonnées ou rien du tout.
  - `metadata` : charge uniquement les métadonnées (durée, etc.).
  - `none` : pas de préchargement. Utile si vous ne voulez pas gaspiller de bande passante sur votre site.



On ne peut pas forcer le préchargement de la musique, c'est toujours le navigateur qui décide. Les navigateurs mobiles, par exemple, ne préchargent jamais la musique pour économiser la bande passante (le temps de chargement étant long sur un portable).

Ajoutons les contrôles et ce sera déjà mieux !

```
<audio src="hype_home.mp3" controls></audio>
```

L'apparence du lecteur audio change en fonction du navigateur. La figure 17.1 représente par exemple le lecteur audio dans Google Chrome.



FIGURE 17.1 – Le lecteur audio dans Google Chrome



Pourquoi ouvrir la balise pour la refermer immédiatement après ?

Cela vous permet d'afficher un message ou de proposer une solution de secours pour les navigateurs qui ne gèrent pas cette nouvelle balise. Par exemple :

```
<audio src="hype_home.mp3" controls>Veuillez mettre à jour
    votre navigateur !</audio>
```

Ceux qui ont un navigateur récent ne verront pas le message. Les anciens navigateurs, qui ne comprennent pas la balise, afficheront en revanche le texte qui se trouve à l'intérieur.



Je vous conseille de proposer une solution de secours en Flash, comme `Dewplayer`. Vous placerez le code correspondant à Flash entre les balises `<audio>` et `</audio>` : ainsi, les anciens navigateurs afficheront le lecteur Flash, tandis que les nouveaux afficheront le lecteur natif.


Télécharger Dewplayer
Code web : 192243



On a vu que certains navigateurs ne géraient pas le MP3, comment faire ?

Il faut proposer plusieurs versions du fichier audio. Dans ce cas, on va construire notre balise comme ceci :

```
<audio controls>
    <source src="hype_home.mp3"></source>
    <source src="hype_home.ogg"></source>
</audio>
```

Le navigateur prendra automatiquement le format qu'il reconnaît.

Insertion d'une vidéo



La balise `<video>` que nous allons découvrir est reconnue par tous les navigateurs récents, y compris Internet Explorer à partir de la version 9 (IE9).

Il suffit d'une simple balise `<video>` pour insérer une vidéo dans la page :

```
<video src="sintel.webm"></video>
```

Mais, là encore, vous risquez d'être déçus si vous utilisez seulement ce code. Aucun contrôle ne permet de lancer la vidéo !

Rajoutons quelques attributs (la plupart sont les mêmes que pour la balise `<audio>`) :

- `poster` : image à afficher à la place de la vidéo tant que celle-ci n'est pas lancée. Par défaut, le navigateur prend la première image de la vidéo mais, comme il s'agit souvent d'une image noire ou d'une image peu représentative de la vidéo, je vous conseille d'en créer une ! Vous pouvez tout simplement faire une capture d'écran d'un moment de la vidéo.
- `controls` : pour ajouter les boutons « Lecture », « Pause » et la barre de défilement. Cela peut sembler indispensable, mais certains sites web préfèrent créer eux-mêmes leurs propres boutons et commander la lecture avec du JavaScript. En ce qui nous concerne, ce sera largement suffisant !
- `width` : pour modifier la largeur de la vidéo.
- `height` : pour modifier la hauteur de la vidéo.
- `loop` : la vidéo sera jouée en boucle.
- `autoplay` : la vidéo sera jouée dès le chargement de la page. Là encore, évitez d'en abuser, c'est en général irritant d'arriver sur un site qui lance quelque chose tout seul !
- `preload` : indique si la vidéo peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
  - `auto` (par défaut) : le navigateur décide s'il doit précharger toute la vidéo, uniquement les métadonnées ou rien du tout.
  - `metadata` : charge uniquement les métadonnées (durée, dimensions, etc.).
  - `none` : pas de préchargement. Utile si vous souhaitez éviter le gaspillage de bande passante sur votre site.



On ne peut pas forcer le préchargement de la vidéo, c'est toujours le navigateur qui décide.



Les proportions de la vidéo sont toujours conservées. Si vous définissez une largeur et une hauteur, le navigateur fera en sorte de ne pas dépasser les dimensions indiquées mais il conservera les proportions.

Voici un code un peu plus complet :

```
<video src="sintel.webm" controls poster="sintel.jpg" width="600"></video>
```

Et le résultat à la figure 17.2.



Pourquoi ouvrir et refermer immédiatement après la balise ?

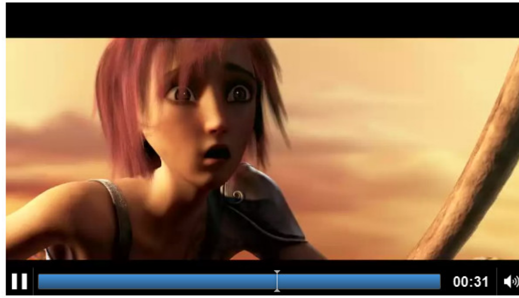


FIGURE 17.2 – Une vidéo avec les options de lecture et une taille définie

La réponse est la même que pour la balise <audio>. Cela vous permet d'afficher un message ou d'utiliser une technique de secours (en Flash) si le navigateur ne reconnaît pas la balise :

```
<video src="sintel.webm" controls poster="sintel.jpg" width="
   600">
    Il est temps de mettre à jour votre navigateur !
</video>
```



Comment contenter tous les navigateurs, puisque chacun reconnaît des formats vidéo différents ?

Vous utiliserez la balise <source> à l'intérieur de la balise <video> pour proposer différents formats. Le navigateur prendra celui qu'il reconnaît :

```
<video controls poster="sintel.jpg" width="600">
    <source src="sintel.mp4" />
    <source src="sintel.webm" />
    <source src="sintel.ogv" />
</video>
```



Les iPhone, iPad et iPod ne reconnaissent à l'heure actuelle que le format H.264 (fichier .mp4)... et uniquement si celui-ci apparaît en premier dans la liste ! Je vous recommande donc d'indiquer le format H.264 en premier pour assurer une compatibilité maximale.



Comment afficher la vidéo en plein écran ?

Ce n'est pas possible à l'heure actuelle. En fait, il existe bien un moyen sous Firefox mais il est un peu caché : il faut faire un clic droit sur la vidéo, puis sélectionner « Plein

écran ».

Il n'y a pas de moyen de forcer le plein écran, même en JavaScript. Cela peut se comprendre, car des sites pourraient perturber fortement la navigation des visiteurs en affichant des vidéos en plein écran sans leur demander leur accord !



Comment protéger ma vidéo, je ne veux pas qu'on puisse la copier facilement !

Ce n'est pas possible. Les balises n'ont pas été conçues pour limiter ou empêcher le téléchargement. C'est assez logique quand on y pense : pour que le visiteur puisse voir la vidéo, il faut bien de toute façon qu'il la télécharge d'une manière ou d'une autre !

N'espérez donc pas empêcher le téléchargement de votre vidéo avec cette technique.



Les lecteurs vidéo Flash permettent de « protéger » le contenu des vidéos mais, là encore, des solutions de contournement existent. De nombreux *plug-ins* permettent de télécharger les vidéos, de Youtube par exemple.

En résumé

- Insérer de la musique ou de la vidéo n'était pas possible autrefois en HTML. Il fallait recourir à un plugin comme Flash.
- Depuis HTML5, les balises `<audio>` et `<video>` ont été introduites et permettent de jouer de la musique et des vidéos sans plugin.
- Il existe plusieurs formats audio et vidéo. Il faut notamment connaître :
  - pour l'audio : MP3 et Ogg Vorbis ;
  - pour la vidéo : H.264, Ogg Theora et WebM.
- Aucun format n'est reconnu par l'ensemble des navigateurs : il faut proposer différentes versions de sa musique ou de sa vidéo pour satisfaire tous les navigateurs.
- Il faut ajouter l'attribut `controls` aux balises `<audio>` et `<video>` pour permettre au visiteur de lancer ou d'arrêter le média.
- Ces balises ne sont pas conçues pour empêcher le téléchargement de la musique et de la vidéo. Vous ne pouvez pas protéger votre média contre la copie.

Chapitre 18

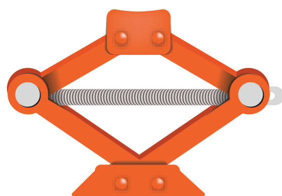
Mise en page adaptative avec les Media Queries

Difficulté :

Savez-vous quelle est la première préoccupation des webmasters qui mettent en place le design de leur site ? La résolution d'écran de leurs visiteurs. Eh oui : selon les écrans, il y a plus ou moins de place, plus ou moins de pixels de largeur.

Cette information est importante lorsque vous construisez un design : comment votre site doit-il s'afficher en fonction des différentes résolutions d'écran ? Si vous avez un écran large, vous risquez d'oublier que certaines personnes naviguent avec des écrans plus petits. Et je ne vous parle même pas des navigateurs des smartphones, qui sont encore moins larges.

C'est là que les *media queries* entrent en jeu. Ce sont des règles à appliquer pour changer le design d'un site en fonction des caractéristiques de l'écran ! Grâce à cette technique, nous pourrons créer un design qui s'adapte automatiquement à l'écran de chaque visiteur !



Mise en place des media queries

Les media queries font partie des nouveautés de CSS3. Il ne s'agit pas de nouvelles propriétés mais de *règles* que l'on peut appliquer dans certaines conditions. Concrètement, vous allez pouvoir dire « Si la résolution de l'écran du visiteur est inférieure à tant, alors applique les propriétés CSS suivantes ». Cela vous permet de changer l'apparence du site dans certaines conditions : vous pourrez augmenter la taille du texte, changer la couleur de fond, positionner différemment votre menu dans certaines résolutions, etc. Contrairement à ce qu'on pourrait penser, les media queries ne concernent pas que les résolutions d'écran. Vous pouvez changer l'apparence de votre site en fonction d'autres critères comme le type d'écran (smartphone, télévision, projecteur...), le nombre de couleurs, l'orientation de l'écran (portrait ou paysage), etc. Les possibilités sont très nombreuses !



Les media queries fonctionnent sur tous les navigateurs récents, y compris Internet Explorer à partir de la version 9 (IE9).

Appliquer une media query

Les media queries sont donc des règles qui indiquent quand on doit appliquer des propriétés CSS. Il y a deux façons de les utiliser :

- en chargeant une feuille de style `.css` différente en fonction de la règle (ex : « Si la résolution est inférieure à 1280px de large, charge le fichier `petite_resolution.css` ») ;
- en écrivant la règle directement dans le fichier `.css` habituel (ex : « Si la résolution est inférieure à 1280px de large, charge les propriétés CSS ci-dessous »).

Chargement d'une feuille de style différente

Vous vous souvenez de la balise `<link />` qui permet, dans notre code HTML, de charger un fichier `.css` ?

```
<link rel="stylesheet" href="style.css" />
```

On peut lui ajouter un attribut `media`, dans lequel on va écrire la règle qui doit s'appliquer pour que le fichier soit chargé. On dit qu'on fait une « requête de media » (*media query* en anglais). Voici un exemple :

```
<link rel="stylesheet" media="screen and (max-width: 1280px)"
    href="petite_resolution.css" />
```

Au final, votre code HTML pourrait proposer plusieurs fichiers CSS : un par défaut (qui est chargé dans tous les cas) et un ou deux autres qui seront chargés en supplément uniquement si la règle correspondante s'applique.

```
<!DOCTYPE html>
```

```html
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" href="style.css" /> <!-- Pour
           tout le monde -->
        <link rel="stylesheet" media="screen and (max-width:
           1280px)" href="petite_resolution.css" /> <!-- Pour
           ceux qui ont une résolution inférieure à 1280px -->
        <title>Media queries</title>
    </head>
```

Chargement des règles directement dans la feuille de style

Une autre technique, que je préfère personnellement pour des raisons pratiques, consiste à écrire ces règles dans le même fichier CSS que d'habitude. Dans ce cas, on écrit la règle dans le fichier `.css` comme ceci :

```css
@media screen and (max-width: 1280px)
{
    /* Rédigez vos propriétés CSS ici */
}
```

Les règles disponibles

Il existe de nombreuses règles permettant de construire des media queries. Je vous présente ici les principales :

- `color` : gestion de la couleur (en bits/pixel).
- `height` : hauteur de la zone d'affichage (fenêtre).
- `width` : largeur de la zone d'affichage (fenêtre).
- `device-height` : hauteur du périphérique.
- `device-width` : largeur du périphérique.
- `orientation` : orientation du périphérique (portrait ou paysage).
- `media` : type d'écran de sortie. Quelques-unes des valeurs possibles :
  - `screen` : écran « classique » ;
  - `handheld` : périphérique mobile ;
  - `print` : impression ;
  - `tv` : télévision ;
  - `projection` : projecteur ;
  - `all` : tous les types d'écran.



On peut rajouter le préfixe `min-` ou `max-` devant la plupart de ces règles. Ainsi, `min-width` signifie « Largeur minimale », `max-height` « Hauteur maximale », etc. La différence entre `width` et `device-width` se perçoit surtout sur les navigateurs mobiles des smartphones, nous en reparlerons plus loin.

Les règles peuvent être combinées à l'aide des mots suivants :

- `only` : « uniquement » ;
- `and` : « et » ;
- `not` : « non ».

Voici quelques exemples de media queries pour vous aider à bien comprendre le principe.

```
/* Sur les écrans, quand la largeur de la fenêtre fait au maximum 1280px */
@media screen and (max-width: 1280px)

/* Sur tous types d'écran, quand la largeur de la fenêtre est comprise entre 1024px et 1280px */
@media all and (min-width: 1024px) and (max-width: 1280px)

/* Sur les téléviseurs */
@media tv

/* Sur tous types d'écrans orientés verticalement */
@media all and (orientation: portrait)
```



Les anciens navigateurs, notamment IE6, IE7 et IE8, ne connaissent pas les media queries mais sont capables d'interpréter le début de la règle (ils savent lire `@media screen` par exemple). Ils vont donc lire les propriétés CSS qui suivent même s'ils ne sont pas concernés par la règle ! Pour éviter cela, une astuce consiste à utiliser le mot-clé `only` que ces vieilles versions ne connaissent pas : « `@media only screen` » ne provoquera pas de bug sur les vieux navigateurs.

Tester les media queries

Les media queries sont surtout utilisées pour adapter le design du site aux différentes largeurs d'écran.

Faisons un test tout simple : nous allons changer la couleur et la taille du texte si la fenêtre fait plus ou moins de 1024 pixels de large. Pour ce test, je vais utiliser la seconde méthode qui consiste à écrire la règle directement dans le même fichier `.css` que d'habitude :

```
/* Paragraphes en bleu par défaut */
p
{
    color: blue;
}

/* Nouvelles règles si la fenêtre fait au plus 1024px de large */
@media screen and (max-width: 1024px)
```

```
{
    p
    {
        color: red;
        background-color: black;
        font-size: 1.2em;
    }
}
```

Dans notre feuille CSS, nous avons d'abord demandé à ce que le texte des paragraphes soit écrit en bleu, jusque là rien de nouveau. En revanche, nous avons ajouté une media query qui s'applique à tous les écrans dont la largeur ne dépasse pas 1024px. À l'intérieur, nous avons appliqué des règles CSS sur les paragraphes pour les écrire plus gros et en rouge.

Résultat : la page n'a pas la même apparence selon la taille de la fenêtre (figure 18.1) ! Essayez de la redimensionner pour voir !

▷ Tester ce code
Code web : 358915

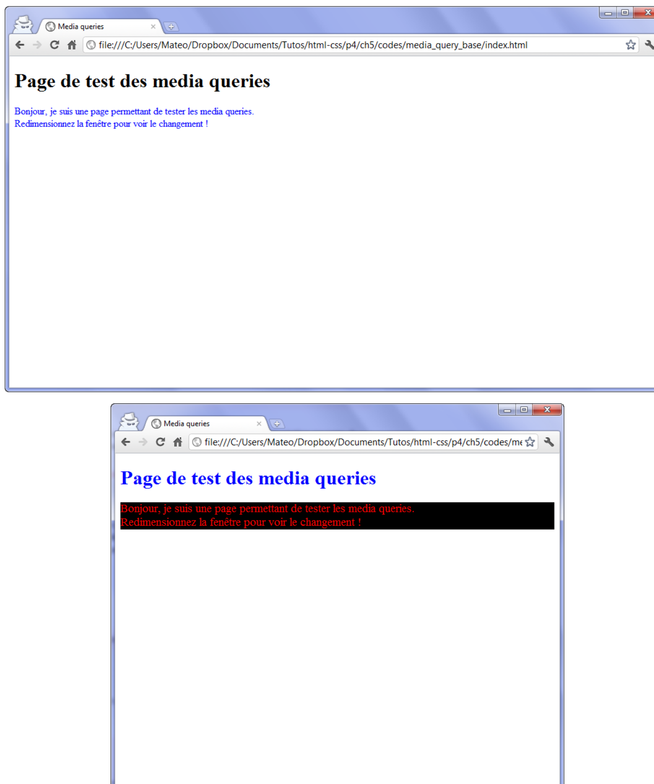


Figure 18.1 – L'apparence du texte change en fonction de la taille de la fenêtre

Mise en pratique des media queries sur le design

Bon, changer la couleur du texte, c'est bien joli mais cela n'apporte pas grand-chose. Par contre, cela devient de suite plus intéressant quand on se sert des media queries pour modifier l'apparence de son site en fonction de la résolution. Vous allez voir qu'on peut faire tout ce qu'on veut !

Pour cet exemple, je vous propose de reprendre le design que nous avons créé pour le site web de Zozor (figure 18.2).

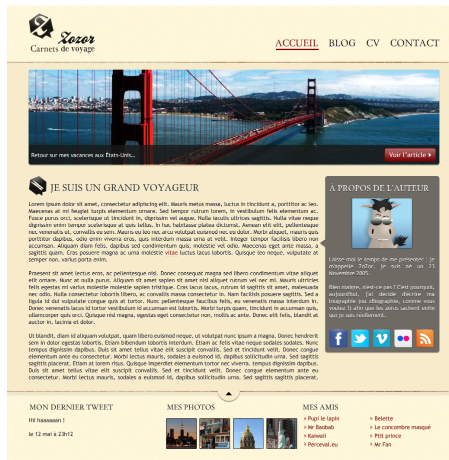


FIGURE 18.2 – Le site web réalisé lors du TP

Le site est bien adapté à la plupart des résolutions d'écran mais, quand l'écran est plus petit que 1024 px, il devient nécessaire de « scroller » vers la droite pour voir toute la page. Le site n'est donc pas très pratique à consulter sur un petit écran.

Je vous propose d'utiliser les media queries pour changer l'apparence du site sur les résolutions inférieures à 1024 px de largeur. Nous allons opérer les modifications suivantes :

- le menu de navigation en haut à droite sera disposé en hauteur plutôt qu'en largeur, et les liens seront écrits en plus petit ;
- la bannière avec le pont de San Francisco (le *Golden Gate*) sera supprimée, car elle prend beaucoup de place et n'apporte pas beaucoup d'informations ;
- le bloc `<aside>` « À propos de l'auteur » sera placé sous l'article (et non pas à côté), et son contenu sera réorganisé (la photo de Zozor sera positionnée en flottant).

On pourrait bien entendu faire beaucoup d'autres modifications : changer la couleur, la disposition du pied de page, etc. Mais cela sera déjà bien suffisant pour nous entraîner avec les media queries.

Nous allons travailler directement à l'intérieur du fichier `style.css` que nous avons réalisé lors du TP. Nous y ajouterons quelques instructions media queries pour adapter le design. Je vous invite à télécharger les fichiers du TP si vous ne les avez pas déjà.

▷ Télécharger le TP
Code web : 975009

La page

Pour le moment, la largeur de la page est fixée à 900 px et le contenu est centré :

```
#bloc_page
{
    width: 900px;
    margin: auto;
}
```

À la suite de ces lignes, je vous propose d'ajouter la règle media query suivante :

```
@media all and (max-width: 1024px)
{
    #bloc_page
    {
        width: auto;
    }
}
```

La règle signifie : « Pour tous les types d'écrans, si la largeur de la fenêtre ne dépasse pas 1024 px, alors exécuter les règles CSS suivantes ».

Les règles CSS en question sont très simples, il n'y en a en fait qu'une seule : on donne une largeur automatique à la page (plutôt qu'une largeur fixe de 900 px). La page prendra alors tout l'espace disponible dans la fenêtre. Cela évite l'apparition de barres de défilement horizontales sur les petites résolutions.



`auto` est la valeur par défaut de la propriété `width`. Par défaut, les blocs ont une largeur automatique (ils prennent toute la place disponible). Cette valeur « écrase » celle que nous avions forcée à 900px quelques lignes plus haut : nous revenons donc au comportement par défaut du bloc.

Le menu de navigation

Nous voulons que le menu de navigation prenne moins de place sur les petites résolutions. Plutôt que de lui donner une dimension fixe, nous allons lui redonner sa dimension automatique flexible d'origine. Chaque élément du menu s'écrira en dessous du précédent : pour cela, nous transformerons les puces en `block` plutôt qu'en `inline-block`.

Enfin, le texte sera écrit plus petit et nous retirons la bordure en bas des liens lors du survol, car elle est moins adaptée à cette disposition.

```
@media all and (max-width: 1024px)
{
```

```
    nav
    {
        width: auto;
        text-align: left;
    }

    nav li
    {
        display: block;
        padding-left: 4px;
    }

    nav a
    {
        font-size: 1.1em;
    }

    nav a:hover
    {
        border-bottom: 0;
    }
}
```

La bannière

Pour retirer la bannière, rien de plus simple : nous utilisons la propriété `display` à laquelle nous affectons la valeur `none`. Si la fenêtre est trop petite, nous préférons masquer complètement la bannière :

```
@media all and (max-width: 1024px)
{
    #banniere_image
    {
        display: none;
    }
}
```

Le bloc « À propos de l'auteur »

Plutôt que de placer ce bloc à droite de l'article, nous allons le faire passer en-dessous. Ce type de disposition « de haut en bas » est plus adapté aux petits écrans.

À l'intérieur du bloc, nous réajustons un peu la position des éléments : la photo de Zozor, notamment, sera placée en flottant à droite.

```
@media all and (max-width: 1024px)
{
    article, aside
```

```
    {
        width: auto;
        display: block;
        margin-bottom: 15px;
    }

    #fleche_bulle
    {
        display: none;
    }

    #photo_zozor img
    {
        width: 110px;
        float: right;
        margin-left: 15px;
    }

    aside p:last-child
    {
        text-align: center;
    }
}
```



Que signifie `aside p:last-child` ?

C'est un sélecteur avancé que nous n'avons pas utilisé jusqu'ici. `aside p` signifie « Tous les paragraphes à l'intérieur de la balise `<aside>` ». Avec `:last-child`, on cible uniquement le dernier paragraphe dans le bloc `aside` (celui qui contient les liens vers Facebook et Twitter), pour pouvoir centrer les images. Bien entendu, on aurait aussi pu affecter une `class` ou un `id` à ce paragraphe pour le cibler directement, mais je n'ai pas voulu modifier le code HTML.

Le résultat

La page est désormais complètement réorganisée lorsque la fenêtre fait 1024 px ou moins de largeur. Regardez par vous-mêmes le résultat, la figure 18.3 parle d'elle-même !

▷ Tester la page
Code web : 865167



FIGURE 18.3 – Le même site, présenté différemment en fonction de la largeur de l'écran

Media queries et navigateurs mobiles

Comme vous le savez sûrement, les écrans des smartphones sont beaucoup moins larges que nos écrans habituels (seulement quelques centaines de pixels de large). Pour s'adapter, les navigateurs mobiles affichent le site en « dézoomant », ce qui permet d'avoir un aperçu de l'ensemble de la page. La zone d'affichage simulée est appelée le **viewport** : c'est la largeur de la fenêtre du navigateur sur le mobile.

En CSS, avec les media queries, si vous ciblez l'écran avec `max-width` sur un mobile, celui-ci va comparer la largeur que vous indiquez avec celle de son viewport. Le problème, c'est que le viewport change selon le navigateur mobile utilisé !

Navigateur	**Largeur du *viewport* par défaut**
Opera Mobile	850 pixels
iPhone Safari	980 pixels
Android	800 pixels
Windows Phone 7	1024 pixels

Un iPhone se comporte comme si la fenêtre faisait 980 px de large, tandis qu'un Android se comporte comme si la fenêtre faisait 800 px !

Pour cibler les smartphones, plutôt que d'utiliser `max-width`, il peut être intéressant de recourir à `max-device-width` : c'est la largeur du périphérique. Les périphériques mobiles ne dépassant pas 480 px de large, on pourra viser uniquement les navigateurs mobiles avec cette media query :

```
@media all and (max-device-width: 480px)
{
    /* Vos règles CSS pour les mobiles ici */
}
```



Pourquoi ne pas cibler les mobiles avec la règle media `handheld` ?

Je vois que vous suivez, très bien ! En effet, on peut (en théorie) cibler les écrans mobiles avec le media `handheld`... Malheureusement, aucun navigateur mobile à part Opera mobile ne reconnaît `handheld`. Ils se comportent tous comme s'ils étaient des écrans normaux (`screen`). On ne peut donc pas vraiment utiliser `handheld` pour viser les mobiles.

Vous pouvez modifier la largeur viewport du navigateur mobile avec une balise `meta` à insérer dans l'en-tête (`<head>`) du document :

```
<meta name="viewport" content="width=320" />
```

Vous pouvez utiliser cette balise pour modifier la façon dont le contenu de votre page s'organise sur les mobiles. Pour obtenir un rendu facile à lire, sans zoom, vous pouvez demander à ce que le viewport soit le même que la largeur de l'écran :

```
<meta name="viewport" content="width=device-width" />
```

En résumé

- Les media queries permettent de charger des styles CSS différents en fonction de certains paramètres.
- Les paramètres autorisés par les media queries sont nombreux : nombre de couleurs, résolution de l'écran, orientation... En pratique, on s'en sert surtout pour modifier l'apparence du site en fonction des différentes résolutions d'écran.
- On crée une media query avec la directive `@media` suivie du type d'écran et d'une ou plusieurs conditions (comme la largeur maximale d'écran). Le style CSS qui suit sera activé uniquement si les conditions sont remplies.
- Les navigateurs mobiles simulent une largeur d'écran : on appelle cela le *viewport*.
- On peut cibler les smartphones grâce à une règle basée sur le nombre réel de pixels affichés à l'écran : `max-device-width`.

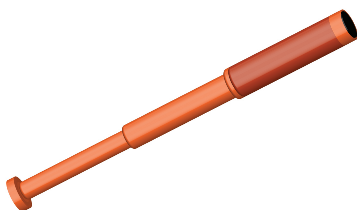
Chapitre 19

Aller plus loin

Difficulté :

Alors que ce livre touche à sa fin, la tentation est grande de penser que l'on a tout vu. Tout vu ? Vous n'avez quand même pas cru cela ? Allons bon, il vous reste des centaines de choses à découvrir, que ce soit sur HTML, CSS, ou les technologies qui y sont liées (PHP, JavaScript. . .).

Ce chapitre a pour but de vous donner quelques directions pour compléter votre apprentissage. Alors ne soyez pas tristes, car vous n'avez pas fini de faire des découvertes !



Du site web à l'application web (JavaScript, AJAX...)

JavaScript est un langage qui existe depuis de nombreuses années maintenant et que l'on utilise fréquemment sur le Web en plus de HTML et CSS. C'est probablement l'un des premiers langages que vous voudrez apprendre maintenant que vous avez des connaissances en HTML et CSS.



À quoi JavaScript peut-il bien servir ? On ne peut pas tout faire avec HTML et CSS ?

On peut faire déjà beaucoup de choses en HTML et CSS mais, lorsqu'on veut rendre sa page plus interactive, un langage comme JavaScript devient indispensable.

Voici quelques exemples de ce à quoi peut servir JavaScript :

- On l'utilisera le plus souvent pour modifier des propriétés CSS sans avoir à recharger la page. Par exemple, vous pointez sur une image et le fond de votre site change de couleur (ce n'est pas possible à faire avec un `:hover` car cela concerne deux balises différentes, c'est bien là une limite du CSS).
- On peut l'utiliser aussi pour modifier le code source HTML sans avoir à recharger la page, *pendant* que le visiteur consulte la page.
- Il permet aussi d'afficher des boîtes de dialogue à l'écran du visiteur...
- ... ou encore de modifier la taille de la fenêtre.

JavaScript est un langage qui se rapproche des langages de programmation tels que le C, C++, Python, Ruby... À l'inverse, HTML et CSS sont davantage des langages de description : ils décrivent comment la page doit apparaître mais ils ne donnent pas d'ordres directs à l'ordinateur (« fais ceci, fais cela... »), contrairement à JavaScript.



JavaScript n'a *aucun* rapport avec le langage Java. Seuls les noms se ressemblent.

JavaScript est régulièrement utilisé aujourd'hui pour faire de l'AJAX (*Asynchronous JavaScript And XML*). Cette technique permet de modifier une partie de la page web que le visiteur consulte en échangeant des données avec le serveur. Cela donne l'impression que les pages sont plus dynamiques et plus réactives. Le visiteur n'a plus besoin de recharger systématiquement toute la page.

Les navigateurs sont de plus en plus efficaces dans leur traitement de JavaScript, ce qui fait que les pages qui utilisent JavaScript sont de plus en plus réactives. On peut ainsi arriver aujourd'hui à créer des sites qui deviennent littéralement des applications web, l'équivalent de logiciels mais disponibles sous forme de sites web ! Un exemple célèbre : Google Docs, la suite bureautique de Google, disponible sur le Web (figure 19.1).

Pour en savoir plus sur JavaScript, lisez le tutoriel du Site du Zéro !

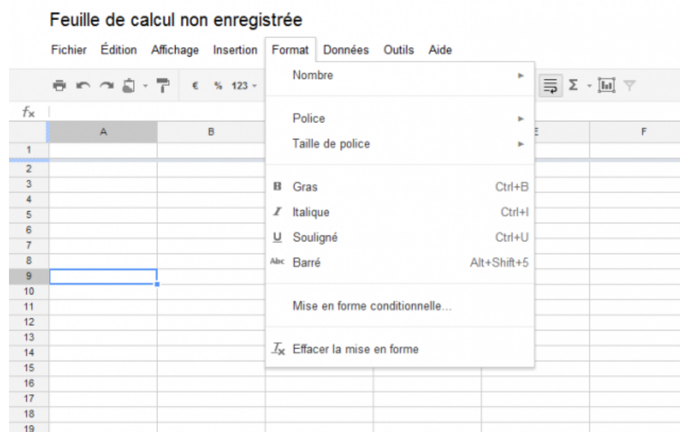


FIGURE 19.1 – Le tableur Google Docs

▷ Lire le tutoriel JavaScript
Code web : 893076

Technologies liées à HTML5 (Canvas, SVG, Web Sockets...)

Le W3C ne travaille pas que sur les langages HTML et CSS. Ce sont certes les plus connus, mais le W3C cherche aussi à définir d'autres technologies qui viennent compléter HTML et CSS. Elles sont nombreuses et on les confond d'ailleurs souvent avec HTML5.



En fait, HTML5 est devenu un mot très utilisé qui fait référence à d'autres technologies que HTML. Quand quelqu'un vous parle de « HTML5 » aujourd'hui, il fait peut-être aussi référence à d'autres éléments qui sortent du cadre strict du HTML.

Voici une petite liste de ces nouvelles technologies introduites en parallèle de HTML5 (notez que certaines ne sont pas vraiment « nouvelles » mais elles reviennent sur le devant de la scène) :

- **Canvas** : permet de dessiner au sein de la page web, à l'intérieur de la balise HTML `<canvas>`. On peut dessiner des formes (triangles, cercles...) mais aussi ajouter des images, les manipuler, appliquer des filtres graphiques... Au final, cela nous permet de réaliser aujourd'hui de véritables jeux et des applications graphiques directement dans des pages web ! Vous pouvez par exemple consulter le tutoriel du Site du Zéro sur Canvas (code web : 510274).
- **SVG** : permet de créer des dessins vectoriels au sein des pages web. À la différence

de Canvas, ces dessins peuvent être agrandis à l'infini (c'est le principe du vectoriel). Le logiciel `Inkscape` est connu pour permettre de dessiner des SVG. Il existe un tutoriel sur le Site du Zéro sur SVG (code web : 678000).
- **Drag & Drop** : permet de faire « glisser-déposer » des objets dans la page web, de la même façon qu'on peut faire glisser-déposer des fichiers sur son bureau. Gmail l'utilise pour permettre d'ajouter facilement des pièces jointes à un e-mail.
- **File API** : permet d'accéder aux fichiers stockés sur la machine du visiteur (avec son autorisation). On l'utilisera notamment en combinaison avec le Drag & Drop.
- **Géolocalisation** : pour localiser le visiteur et lui proposer des services liés au lieu où il se trouve (ex. : les horaires des salles de cinéma proches). La localisation n'est pas toujours très précise, mais cela peut permettre de repérer un visiteur à quelques kilomètres près (avec son accord).
- **Web Storage** : permet de stocker un grand nombre d'informations sur la machine du visiteur. C'est une alternative plus puissante aux traditionnels cookies. Les informations sont hiérarchisées, comme dans une base de données.
- **Appcache** : permet de demander au navigateur de mettre en cache certains fichiers, qu'il ne cherchera alors plus à télécharger systématiquement. Très utile pour créer des applications web qui peuvent fonctionner même en mode « hors ligne » (déconnecté).
- **Web Sockets** : permet des échanges plus rapides, en temps réel, entre le navigateur du visiteur et le serveur qui gère le site web (c'est une sorte d'AJAX amélioré). C'est un peu l'avenir des applications web, qui pourront devenir aussi réactives que les vrais programmes.
- **WebGL** : permet d'introduire de la 3D dans les pages web, en utilisant le standard de la 3D OpenGL (figure 19.2). Les scènes 3D sont directement gérées par la carte graphique.



La plupart de ces technologies s'utilisent avec JavaScript. Il s'agit donc de nouvelles fonctionnalités que l'on peut utiliser en JavaScript.

Comme vous le voyez, vous avez de nouveaux mondes à découvrir ! Dès que vous connaîtrez suffisamment JavaScript, vous pourrez aller encore plus loin dans la gestion de votre site web... que vous pourrez même transformer en véritable application !

Les sites web dynamiques (PHP, JEE, ASP .NET...)

Les langages dont nous allons parler ici sont eux aussi des langages de programmation. Comme JavaScript ? Oui, mais avec une différence importante : JavaScript s'exécute sur la machine de vos visiteurs, tandis que les langages que nous allons voir s'exécutent sur le « serveur » qui contient votre site web.



Quelle différence cela fait-il que le programme tourne sur la machine du visiteur ou sur le serveur ?



Figure 19.2 – Une application web 3D utilisant WebGL

Les différences sont importantes. Tout d'abord, en termes de puissance, un serveur sera bien souvent plus rapide que la machine de vos visiteurs, ce qui permet d'effectuer des calculs plus complexes. Vous avez aussi davantage de contrôle côté serveur qu'en JavaScript... Mais le JavaScript reste irremplaçable car il y a certaines actions que vous ne pouvez faire que du côté « visiteur ».

Les langages serveur permettent de générer la page web lorsque le visiteur arrive sur votre site (figure 19.3). Chaque visiteur peut donc obtenir une page web personnalisée suivant ses besoins !

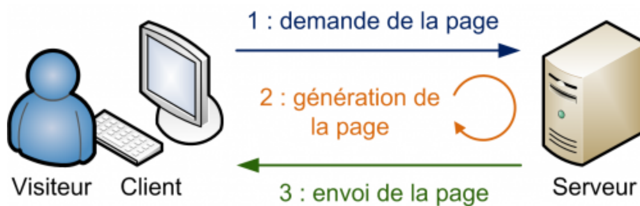


Figure 19.3 – Échange de données avec un serveur

Les langages ne servent donc pas aux mêmes choses, mais ils se complètent. Si vous combinez HTML + CSS + JavaScript + PHP, par exemple, vous pouvez faire de l'AJAX (échanges de données entre la page et le serveur), vous pouvez effectuer des calculs, stocker des informations dans des bases de données... bref, faire de vrais sites web dynamiques !

Les langages « côté serveur » sont nombreux. Citons-en quelques-uns :

– **PHP** : l'un des plus connus. Facile à utiliser et puissant, il est utilisé notamment par Facebook... et le Site du Zéro. J'ai d'ailleurs rédigé un tutoriel sur PHP sur le

Site du Zéro, que vous pouvez lire en suivant le code web ci-dessous.
- **JEE** (Java) : très utilisé dans le monde professionnel, il s'agit d'une extension du langage Java qui permet de réaliser des sites web dynamiques, puissants et robustes. Au début, il est un peu plus complexe à prendre en main que PHP.
- **ASP .NET** (C#) : assez semblable à JEE, c'est le langage de Microsoft. On l'utilise en combinaison avec d'autres technologies Microsoft (Windows Server...). Il utilise le puissant framework .NET, véritable couteau suisse des développeurs, qui offre de nombreuses fonctionnalités.
- **Django** (Python) : une extension du langage Python qui permet de réaliser rapidement et facilement des sites web dynamiques. Il est connu pour générer des interfaces d'administration prêtes à l'emploi.
- **Ruby on Rails** (Ruby) : une extension du langage Ruby, assez similaire à Django, qui permet de réaliser des sites web dynamiques facilement et avec une grande souplesse.

▷ Lire le tutoriel sur PHP
Code web : 955969



Connaître l'un de ces langages est indispensable si vous voulez traiter le résultat des formulaires HTML ! Souvenez-vous de la balise `<form>` : je vous avais expliqué comment créer des formulaires, mais pas comment *récupérer* les informations saisies par vos visiteurs. Il vous faut obligatoirement un langage serveur, comme PHP, pour récupérer et traiter ces données !

Au final, ces langages vous permettent de réaliser vos rêves les plus fous sur votre site web :

- forums ;
- newsletter ;
- compteur de visiteurs ;
- système de news automatisé ;
- gestion de membres ;
- jeux web (jeux de stratégie, élevage d'animaux virtuels...) ;
- etc.



Il est indispensable connaître les langages HTML et CSS avant d'apprendre un langage serveur comme PHP !

Bonne découverte !

Cinquième partie

Annexes

Chapitre 20

Envoyez votre site sur le Web

Difficulté :

Votre site est tout beau, tout propre, tout prêt... mais comme il est sur votre disque dur, personne d'autre ne va pouvoir en profiter !

Vous aimeriez donc l'envoyer sur le Web, mais... vous ne savez pas comment faire. Nous allons découvrir dans cette annexe tout ce qu'il faut savoir pour envoyer son site sur le Web :

1. Comment réserver un **nom de domaine** ?
2. Qu'est-ce qu'un **hébergeur** et comment cela fonctionne-t-il ?
3. Enfin, comment utiliser un **client FTP** pour pouvoir transférer les fichiers sur le Net ?



Le nom de domaine

Savez-vous ce qu'est un **nom de domaine** ? Il s'agit en fait d'une adresse sur le Web : `siteduzero.com` est par exemple un nom de domaine.

Un nom de domaine est constitué de deux parties : `siteduzero.com`.

- En rouge, le nom de domaine proprement dit. Il s'agit d'un nom que l'on peut en général choisir librement, tant que personne ne l'a réservé avant nous. Il peut contenir des lettres et des chiffres, mais pas de symboles particuliers (comme le « ç » français, le « é », le « è », les espaces, etc.).
- En bleu, l'extension (aussi appelée « TLD », de l'anglais *top-level domain*). Il existe grosso modo une extension par pays (`.fr` pour la France, `.be` pour la Belgique, `.ca` pour le Canada, etc.). Toutefois, il y a aussi des extensions utilisées au niveau international comme `.com`, `.net`, `.org`. Elles étaient au départ réservées aux sites commerciaux, aux organisations, . . . mais cela fait longtemps que tout le monde peut les réserver. D'ailleurs, `.com` est très probablement l'extension la plus utilisée sur le Web.



En général, un site web voit son adresse précédée par `www`, comme par exemple `www.siteduzero.com`. Cela ne fait pas partie du nom de domaine : en fait, `www` est ce qu'on appelle un sous-domaine, et on peut en théorie en créer autant qu'on veut une fois qu'on est propriétaire du nom de domaine. Le `www` a été adopté par tous les webmasters, c'est une sorte de convention, mais elle n'est absolument pas obligatoire.

Réserver un nom de domaine



Moi aussi je veux un nom de domaine pour mon site ! Comment dois-je faire ?

Alors j'ai une bonne et une mauvaise nouvelle. Comme d'habitude, on va commencer par la mauvaise :

- **la mauvaise** : ce n'est pas gratuit. . .
- **la bonne** : . . . ce n'est vraiment pas cher du tout.

En effet, un nom de domaine coûte entre 7 et 12 euros par an. Le prix peut varier en fonction de l'extension. Ainsi, l'extension `.info` est généralement proposée à plus bas prix et peut s'avérer être une alternative intéressante. Mais si vous voulez une adresse plus « courante », il faudra plutôt viser une extension de type `.com` ou encore `.fr`.

Pour réserver un nom de domaine, deux solutions :

- Passer par un **registrar** spécialisé. C'est un organisme qui sert d'intermédiaire entre

l'ICANN[1] et vous. 1&1, OVH et Gandi sont de célèbres registrars français.
- Encore mieux : vous pouvez commander le nom de domaine en même temps que l'hébergement (c'est ce que je vous conseille). De cette manière, vous faites d'une pierre deux coups, vu que vous aurez de toute façon besoin de l'hébergement *et* du nom de domaine.

Dans ce chapitre, nous allons voir comment commander un nom de domaine en même temps que l'hébergement, c'est de loin la solution la plus simple et la moins coûteuse pour vous.

L'hébergeur

Intéressons-nous maintenant à l'hébergeur.



Qu'est-ce qu'un hébergeur et pourquoi aurais-je besoin de lui ?

Sur Internet, tous les sites web sont stockés sur des ordinateurs particuliers appelés **serveurs** (figure 20.1). Ce sont des ordinateurs généralement très puissants, qui restent tout le temps allumés. Ils contiennent les pages des sites web et les délivrent aux internautes qui les demandent, à toute heure du jour et de la nuit.

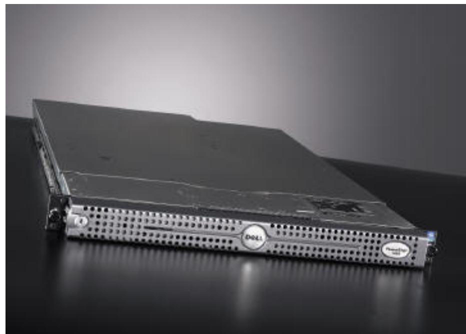


FIGURE 20.1 – Un serveur

Un serveur ne possède pas d'écran car, la plupart du temps, il tourne tout seul sans qu'il y ait besoin de faire quoi que ce soit dessus. Comme vous le voyez, les serveurs sont très plats : c'est un format spécial de serveur (appelé « 1U »). Cela permet de les empiler dans des **baies**[2] (figure 20.2).

Comme vous le voyez, il y a un écran pour toute la baie. C'est suffisant car on ne branche l'écran sur un serveur que si celui-ci rencontre un problème. La plupart du temps, heureusement, le serveur travaille sans broncher.

1. L'ICANN est l'organisation qui gère l'ensemble des noms de domaine au niveau international.
2. Une baie est une sorte d'armoire climatisée pour serveurs.



Figure 20.2 – Une baie de serveurs

Le rôle de l'hébergeur

L'hébergeur est une entreprise qui se charge de gérer des baies de serveurs. Elle s'assure du bon fonctionnement des serveurs 24h/24, 7j/7. En effet, si l'un d'eux tombe en panne, tous les sites présents sur la machine deviennent inaccessibles (et cela fait des clients mécontents).

Ces baies se situent dans des lieux particuliers appelés **datacenters** (figure 20.3). Les datacenters sont donc en quelque sorte des « entrepôts à serveurs » et leur accès est très protégé.



Il est aussi possible, en théorie, d'héberger un site sur son propre ordinateur. Toutefois, c'est complexe : il vaut mieux avoir des connaissances en Linux, l'ordinateur doit être assez puissant, tourner jour et nuit et... surtout... la connexion doit être à très très haut débit (surtout en **upload**, la vitesse d'envoi des fichiers compte énormément). Les particuliers n'ont en règle générale pas une connexion suffisamment puissante pour héberger des sites, contrairement aux datacenters : ceux-ci sont câblés en fibre optique (ce qui permet d'atteindre des vitesses de plusieurs Gbps !)

Bref, gérer un serveur soi-même est complexe et, la plupart du temps, les particuliers et les entreprises font appel à un hébergeur dont c'est le métier.

Trouver un hébergeur

Les hébergeurs, contrairement aux registrars, sont très très nombreux. Il y en a de tous types, à tous les prix. Il y a un vocabulaire à connaître pour vous repérer dans leurs offres :

- **Hébergement mutualisé** : si vous optez pour une offre d'hébergement mutualisé, votre site sera placé sur un serveur gérant plusieurs sites à la fois (peut-être



FIGURE 20.3 – Un datacenter, dans lequel on voit plusieurs baies de serveurs

une centaine, peut-être plus). *C'est l'offre la moins chère et c'est celle que je vous recommande de viser* si vous démarrez votre site web.
- **Hébergement dédié virtuel** : cette fois, le serveur ne gère que très peu de sites (généralement moins d'une dizaine). Cette offre est généralement adaptée aux sites qui d'un côté ne peuvent plus tenir sur un hébergement mutualisé car ils ont trop de trafic (trop de visiteurs), mais qui par ailleurs ne peuvent pas se payer un hébergement dédié (voir ci-dessous).
- **Hébergement dédié** (on parle aussi de « serveur dédié ») : c'est le nec plus ultra. Le serveur gère uniquement votre site et aucun autre. Attention, cela coûte assez cher et il vaut mieux avoir des connaissances en Linux pour administrer le serveur à distance. Par exemple, le Site du Zéro est lui-même sur un hébergement dédié, car son trafic est très important.



Mais où puis-je trouver un hébergeur ?

Oh cela, c'est très simple. Une recherche dans Google de « hébergeur web » vous donnera plusieurs millions de résultats. Vous n'aurez que l'embarras du choix.

Si je puis me permettre un conseil, je vous recommande de jeter un coup d'œil à l'hébergeur **PlanetHoster**, qui propose des services d'hébergement de qualité. Ils font d'ailleurs des réductions pour tous les lecteurs de ce livre !

▷ Les offres de *PlanetHoster*
Code web : 662697

Si les offres de *PlanetHoster* ne vous conviennent pas, vous pouvez regarder chez l'hébergeur *1&1* ou encore *MavenHosting*, qui proposent d'autres offres pour les particuliers et entreprises.

La suite de ce chapitre détaille la procédure pour héberger votre site chez *PlanetHoster*, mais sachez que cela fonctionne quasiment de la même manière avec *1&1* et *MavenHosting*.

Revenons à *PlanetHoster*. Cet hébergeur propose plusieurs offres d'hébergement mutualisé, comme le montre la figure 20.4.

Plan Essentiel	Plan Performance	Plan Illimité
€2.99 /mois	€5.99 /mois	€10.99 /mois
10GB Espace disque	50GB Espace disque	Illimité Espace disque
250GB Trafic	Illimité Trafic	Illimité Trafic
Aucun Frais D'installation	Aucun Frais D'installation	Aucun Frais D'installation
Activation Instantanée	Activation Instantanée	Activation Instantanée
Nom de domaine GRATUIT	Nom de domaine GRATUIT	Nom de domaine GRATUIT
COMMANDER	COMMANDER	COMMANDER
PLUS DE DÉTAILS	PLUS DE DÉTAILS	PLUS DE DÉTAILS

FIGURE 20.4 – Les offres mutualisées de *PlanetHoster*

PlanetHoster fait des réductions spéciales sur tous ses hébergements mutualisés pour les lecteurs de ce livre :

– 5% de remise pour le plan « Essentiel » ;
– 15% de remise pour les plans « Performance » et « Illimité ».

Ces remises sont valables si vous saisissez un code promotionnel (j'en reparle un peu plus bas) pour une commande annuelle de l'un de ces plans :

– **Plan Essentiel** : 10 Go d'espace disque, 250 Go de trafic, un nom de domaine et un nombre illimité de comptes e-mail ;
– **Plan Performance** : 50 Go d'espace disque, trafic illimité, un nom de domaine et un nombre illimité de comptes e-mail ;
– **Plan Illimité** : espace disque, trafic et nombre de comptes e-mail illimités ainsi qu'un nom de domaine.



Tous ces packs donnent la possibilité d'avoir des noms de domaine en `.fr` !

Ces offres sont en fait très similaires, elles diffèrent seulement au niveau de l'espace de stockage et du trafic.



Mais qu'est-ce qu'ils appellent le « trafic » ?

Le trafic, c'est la quantité de données envoyées par mois aux visiteurs de votre site. Par exemple, si vous avez une image de 1 Mo sur votre site et qu'elle est chargée 500 fois dans le mois par vos visiteurs, alors vous créez un trafic de 500 Mo. En pratique, il faut savoir que les navigateurs des visiteurs mettent généralement en cache les images, ce qui leur évite d'avoir à recharger plusieurs fois une même image. Cela diminue d'autant plus le trafic nécessaire.

Si vous avez beaucoup de visiteurs (donc beaucoup de trafic), il faudra choisir un plan qui autorise plus de trafic.

Commander un hébergement pour votre site web

Après avoir cliqué sur n'importe quel bouton « Commander », nous arrivons sur la page représentée à la figure 20.5.

FIGURE 20.5 – Formulaire de commande de *PlanetHoster*

Nous avons ici trois informations :

- **Choisissez un produit** : indique quel plan vous avez choisi. Vous pouvez en changer directement *via* cet encart ;
- **Nom de domaine** : cette partie vous permet de choisir le nom de domaine de votre site web. Nous allons y venir.
- **Résumé** : ce cadre, comme son nom l'indique, résume votre commande avec le plan choisi ainsi que le prix à payer.

Le champ de texte se trouvant dans le cadre **Résumé** (figure 20.6) vous permet de

saisir un code promotionnel :

- `SiteDuZero-Perso` : si vous commandez un **Plan Essentiel** (à 2,99 euros / mois) ;
- `SiteDuZero` : si vous commandez n'importe quel autre plan.

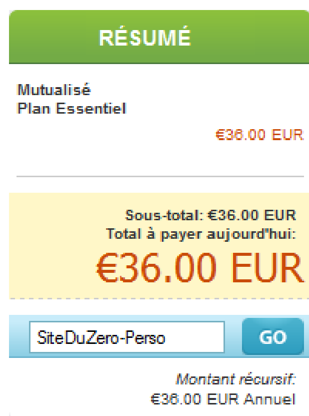


FIGURE 20.6 – Un code promotionnel chez *PlanetHoster*

Validez et vous aurez alors une réduction sur le montant total du pack d'hébergement !

Le champ de texte se trouvant sous **Nom de domaine** vous invite à saisir... votre nom de domaine. Le site de *PlanetHoster* va alors se charger de vérifier si le domaine est disponible. Si c'est bon, vous pouvez passer à la suite. Sinon, il faudra choisir un autre nom de domaine car, quand le domaine est déjà pris, vous ne pouvez pas faire grand chose.

Ensuite, le site vous demande si vous désirez qu'il enregistre ce domaine ou si vous désirez l'enregistrer séparément (figure 20.7).



FIGURE 20.7 – *PlanetHoster* indique si le nom de domaine est déjà enregistré



Obtenir un « vrai » nom de domaine (`.fr`, `.com`, `.net`, `.org`...) est habituellement payant chez les hébergeurs. Néanmoins, si vous achetez un hébergement d'un an chez *PlanetHoster*, le nom de domaine est offert (il est compris dans l'hébergement).

Il ne vous reste plus qu'à renseigner vos coordonnées et finaliser l'achat par carte bancaire ou Paypal. Une fois les formalités et le paiement effectués, vous êtes redirigés vers *PlanetHoster*, qui vous confirme la prise en compte de votre commande. Vous devriez recevoir un peu plus tard un e-mail vous indiquant toutes les informations nécessaires pour mettre en place votre site. Conservez-les précieusement, vous en aurez besoin.

Lorsque vous avez reçu par e-mail vos identifiants pour vous connecter au serveur de votre hébergeur, vous pouvez passer à l'étape suivante : *envoyer votre site web sur le serveur de votre hébergeur* !

Utiliser un client FTP

Installer un client FTP

FTP signifie *File Transfer Protocol* et, pour faire court et simple, c'est le moyen que l'on utilise pour envoyer nos fichiers. Il existe des logiciels permettant d'utiliser le FTP pour transférer vos fichiers sur Internet.

Bien entendu, des logiciels FTP, il en existe des centaines, gratuits, payants, français, anglais, etc. Pour que nous soyons sur la même longueur d'onde, je vais vous proposer celui que j'utilise, qui est gratuit et en français : **FileZilla** (figure 20.8).



FIGURE 20.8 – L'icône du célèbre client FTP FileZilla



Ce logiciel n'a rien à avoir avec Mozilla, si ce n'est qu'il se termine lui aussi par « zilla ». N'allez donc pas croire que je vous force à utiliser des logiciels d'un même éditeur, c'est tout à fait faux. D'ailleurs, vous pouvez utiliser n'importe quel autre logiciel FTP si cela vous chante, cela ne me dérange absolument pas.

Quoiqu'il en soit, je vais vous montrer quelle est la marche à suivre avec FileZilla. Et la première étape, c'est bien entendu de le télécharger !

▷ Télécharger FileZilla
Code web : 667869

Prenez la version correspondant à votre système d'exploitation (Windows, Mac OS X ou Linux).

Je vous fais confiance pour l'installation, elle est toute simple et vous ne devriez pas avoir de problème. Lancez le logiciel, vous devriez voir quelque chose ressemblant à la figure 20.9.

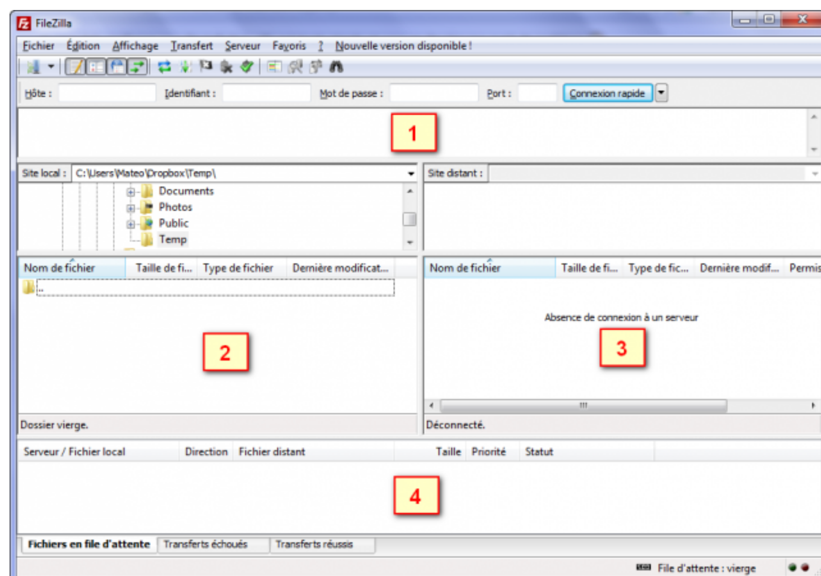


FIGURE 20.9 – FileZilla est ouvert

À première vue, cela semble un peu compliqué (à première vue seulement). En fait, le principe est très simple. Il y a quatre grandes zones à connaître dans la fenêtre :

1. En haut, vous verrez apparaître les messages qu'envoie et reçoit le logiciel. Si vous avez un peu de chance, vous verrez même la machine vous dire bonjour (si si, je vous jure). En général, cette zone ne nous intéresse pas vraiment, sauf s'il y a des messages d'erreur en rouge. . .
2. À gauche, c'est votre disque dur. Dans la partie du haut, vous avez les dossiers et, dans la partie du bas, la liste des fichiers du dossier actuel.
3. À droite, c'est la liste des fichiers envoyés sur le serveur sur Internet. Pour le moment il n'y a rien car on ne s'est pas connecté, mais cela va venir, ne vous en faites pas.
4. Enfin, en bas, vous verrez apparaître les fichiers en cours d'envoi (et le pourcentage d'envoi).

La première étape va être de se connecter au serveur de votre hébergeur.

Configurer le client FTP

Quel que soit l'hébergeur que vous avez choisi, cela fonctionne toujours de la même manière. On va vous fournir *trois informations* qui sont indispensables pour que FileZilla puisse se connecter au serveur :

– **L'IP** : c'est « l'adresse » du serveur. Le plus souvent, on vous donnera une information du type `ftp.mon-site.com`, mais il peut aussi s'agir d'une suite de nombres

comme `122.65.203.27`.
- **Le login** : c'est votre identifiant, on vous probablement demandé de le choisir. Vous avez peut-être mis votre pseudo, ou le nom de votre site. Mon login pourrait par exemple être `mateo21`.
- **Le mot de passe** : soit on vous a demandé de choisir un mot de passe, soit (c'est plus probable) on vous en a attribué un d'office (un truc imprononçable du genre `crf45u7h`).

Si vous avez ces trois informations, vous allez pouvoir continuer. Si vous ne les avez pas, il faut que vous les cherchiez, c'est indispensable. On vous les a probablement envoyées par e-mail. Sinon, n'hésitez pas à les demander à votre hébergeur (IP, login et mot de passe).

Maintenant que nous sommes en possession de ces informations, nous allons les donner à FileZilla, qui en a besoin pour se connecter au serveur.

Cliquez sur la petite icône en haut à gauche (pas sur la petite flèche à droite, mais bien sur l'image), représentée à la figure 20.10.



FIGURE 20.10 – L'icône de connexion de FileZilla

Une fenêtre s'ouvre. Cliquez sur `Nouveau site` et donnez-lui le nom que vous voulez (par exemple « Site du Zéro »). À droite, vous allez devoir indiquer les trois informations dont je viens de vous parler, comme à la figure 20.11.

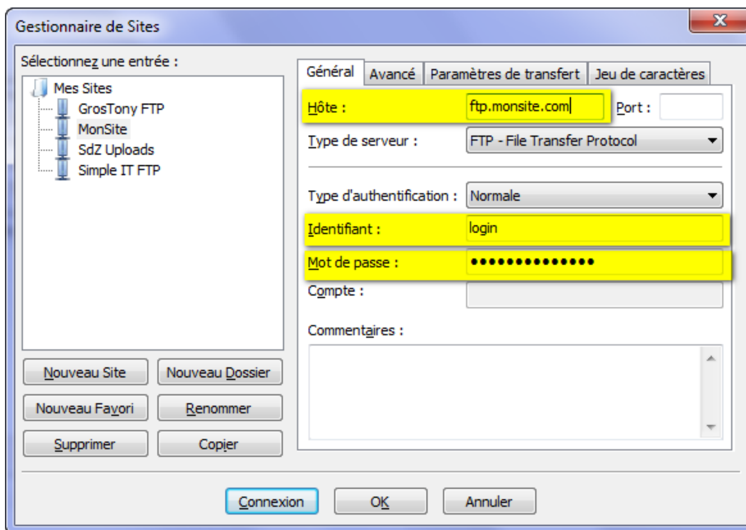


FIGURE 20.11 – Les trois informations à donner à FileZilla

Vous pouvez distinguer en haut l'hôte (c'est là qu'il faut indiquer `ftp.monsite.com`, par exemple). Cochez `Type d'authentification : Normale` pour pouvoir saisir le login et le mot de passe.

Cliquez sur `Connexion` et le tour est (presque) joué.

Transférer les fichiers

À ce stade, deux possibilités :

- Soit la connexion a réussi : vous voyez alors apparaître en haut des messages en vert comme « Connecté ». Dans ce cas, la zone de droite de la fenêtre de FileZilla devrait s'activer et vous verrez les fichiers qui se trouvent déjà sur le serveur (il se peut qu'il y en ait déjà quelques-uns).
- Soit cela a planté, vous avez plein de messages écrits en rouge et là, eh bien… il n'y a pas trente-six solutions : vous vous êtes trompés en tapant l'IP, ou le login, ou le mot de passe. Un de ces éléments est incorrect, veillez à les redemander à votre hébergeur car s'ils sont bons cela *doit* marcher.

Si la connexion a réussi, alors ce que vous avez à faire est très simple : dans la partie de gauche, cherchez où se trouvent, sur votre disque dur, vos fichiers `.html` et `.css` (mais aussi vos images `.jpg`, `.png`, `.gif`, etc.). À gauche, faites un double-clic sur le fichier que vous voulez transférer. Au bout de quelques secondes, il apparaîtra à droite, ce qui voudra dire qu'il a été correctement envoyé sur le serveur, et donc qu'il est accessible sur Internet !



Vous pouvez envoyer n'importe quel type de fichier. Bien entendu, généralement on envoie des fichiers `.php`, `.html`, `.css` et des images, mais vous pouvez aussi très bien envoyer des `.pdf`, des programmes, des `.zip`, etc.

La figure 20.12, par exemple, représente le résultat que l'on obtient après avoir transféré un fichier `index.html` et quelques autres fichiers.

Il apparaît à droite, ce qui veut dire qu'il est maintenant disponible sur le serveur.



Veuillez noter qu'il *faut* que votre page d'accueil s'appelle `index.html`. C'est la page qui sera chargée lorsqu'un nouveau visiteur arrivera sur votre site.

Vous pouvez aussi transférer des dossiers entiers d'un seul coup : il suffit de faire un glisser-déposer du dossier depuis la partie de gauche (ou directement de la fenêtre de votre système d'exploitation) jusqu'à la partie de droite de la fenêtre de FileZilla.

Une fois configuré, vous pouvez voir que l'envoi de fichiers est très simple.

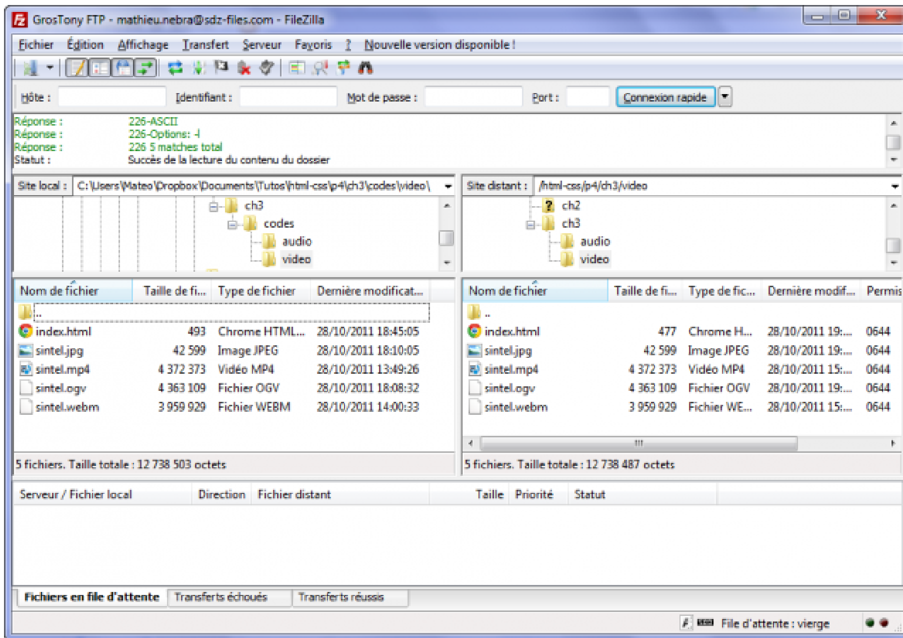


FIGURE 20.12 – Des fichiers sont hébergés sur le FTP

En résumé

- Pour le moment, votre site web n'est visible que par vous, sur votre ordinateur. Il faut l'envoyer sur le Web pour qu'il soit visible par tout le monde.
- Vous avez besoin de deux éléments :
  - Un nom de domaine : c'est l'adresse de votre site web. Vous pouvez réserver une adresse en `.com`, `.fr`, `.net`… Par exemple : `siteduzero.com`.
  - Un hébergeur : c'est lui qui va stocker votre site web sur une machine appelée « serveur ». Son rôle sera d'envoyer votre site à vos visiteurs à toute heure du jour et de la nuit.
- Pour transmettre les fichiers de votre site au serveur de votre hébergeur, il faut utiliser un client FTP comme FileZilla.
- Pour vous connecter au serveur, vous avez besoin de trois informations : l'adresse IP du serveur (ou son nom d'hôte), votre login et votre mot de passe. Ceux-ci vous sont fournis par votre hébergeur.

Chapitre 21

Mémento des balises HTML

Difficulté :

Cette page est une liste *non exhaustive* des balises HTML qui existent. Vous trouverez ici un grand nombre de balises HTML. Nous en avons déjà vu certaines dans le cours, mais il y en a d'autres que nous n'avons pas eu l'occasion d'étudier. Généralement, les balises que nous n'avons pas étudiées sont des balises un peu plus rarement utilisées. Peut-être trouverez-vous votre bonheur dans ce lot de nouvelles balises.

Vous pouvez vous servir de cette annexe comme d'un aide-mémoire lorsque vous développez votre site web.





Attention, j'insiste : *cette annexe n'est pas complète et c'est volontaire*. Je préfère mettre *moins* de balises et garder seulement celles qui me semblent les plus utiles dans la pratique.

Balises de premier niveau

Les balises de premier niveau sont les principales balises qui structurent une page HTML. Elles sont indispensables pour réaliser le « code minimal » d'une page web.

Balise	Description
`<html>`	Balise principale
`<head>`	En-tête de la page
`<body>`	Corps de la page

Code minimal d'une page HTML :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>Titre</title>
    </head>

    <body>

    </body>
</html>
```

Balises d'en-tête

Ces balises sont toutes situées dans l'en-tête de la page web, c'est-à-dire entre `<head>` et `</head>` :

Balise	Description
`<link />`	Liaison avec une feuille de style
`<meta />`	Métadonnées de la page web (charset, mots-clés, etc.)
`<script>`	Code JavaScript
`<style>`	Code CSS
`<title>`	Titre de la page

Balises de structuration du texte

Balise	**Description**
`<abbr>`	Abréviation
`<blockquote>`	Citation (longue)
`<cite>`	Citation du titre d'une œuvre ou d'un évènement
`<q>`	Citation (courte)
`<sup>`	Exposant
`<sub>`	Indice
`<strong>`	Mise en valeur forte
`<em>`	Mise en valeur normale
`<mark>`	Mise en valeur visuelle
`<h1>`	Titre de niveau 1
`<h2>`	Titre de niveau 2
`<h3>`	Titre de niveau 3
`<h4>`	Titre de niveau 4
`<h5>`	Titre de niveau 5
`<h6>`	Titre de niveau 6
`<img />`	Image
`<figure>`	Figure (image, code, etc.)
`<figcaption>`	Description de la figure
`<audio>`	Son
`<video>`	Vidéo
`<source>`	Format source pour les balises `<audio>` et `<video>`
`<a>`	Lien hypertexte
` `	Retour à la ligne
`<p>`	Paragraphe
`<hr />`	Ligne de séparation horizontale
`<address>`	Adresse de contact
`<del>`	Texte supprimé
`<ins>`	Texte inséré
`<dfn>`	Définition
`<kbd>`	Saisie clavier
`<pre>`	Affichage formaté (pour les codes sources)
`<progress>`	Barre de progression
`<time>`	Date ou heure

Balises de listes

Cette section énumère toutes les balises HTML permettant de créer des listes (listes à puces, listes numérotées, listes de définitions...)

Balise	**Description**
`<ul>`	Liste à puces, non numérotée
`<ol>`	Liste numérotée
`<li>`	Élément de la liste à puces
`<dl>`	Liste de définitions
`<dt>`	Terme à définir
`<dd>`	Définition du terme

Balises de tableau

Balise	**Description**
`<table>`	Tableau
`<caption>`	Titre du tableau
`<tr>`	Ligne de tableau
`<th>`	Cellule d'en-tête
`<td>`	Cellule
`<thead>`	Section de l'en-tête du tableau
`<tbody>`	Section du corps du tableau
`<tfoot>`	Section du pied du tableau

Balises de formulaire

Balise	**Description**
`<form>`	Formulaire
`<fieldset>`	Groupe de champs
`<legend>`	Titre d'un groupe de champs
`<label>`	Libellé d'un champ
`<input />`	Champ de formulaire (texte, mot de passe, case à cocher, bouton, etc.)
`<textarea>`	Zone de saisie multiligne
`<select>`	Liste déroulante
`<option>`	Élément d'une liste déroulante
`<optgroup>`	Groupe d'éléments d'une liste déroulante

Balises sectionnantes

Ces balises permettent de construire le squelette de notre site web.

Balise	Description
`<header>`	En-tête
`<nav>`	Liens principaux de navigation
`<footer>`	Pied de page
`<section>`	Section de page
`<article>`	Article (contenu autonome)
`<aside>`	Informations complémentaires

Balises génériques

Les balises génériques sont des balises qui n'ont pas de sens sémantique.

En effet, toutes les autres balises HTML ont un *sens* : `<p>` signifie « Paragraphe », `<h2>` signifie « Sous-titre », etc. Parfois, on a besoin d'utiliser des balises génériques (aussi appelées **balises universelles**) car aucune des autres balises ne convient. On utilise le plus souvent des balises génériques pour construire son design.

Il y a deux balises génériques : l'une est inline, l'autre est block.

Balise	Description
`<span>`	Balise générique de type inline
`<div>`	Balise générique de type block

Ces balises ont un intérêt uniquement si vous leur associez un attribut `class`, `id` ou `style` :

- `class` : indique le nom de la classe CSS à utiliser.
- `id` : donne un nom à la balise. Ce nom doit être unique sur toute la page car il permet d'identifier la balise. Vous pouvez vous servir de l'ID pour de nombreuses choses, par exemple pour créer un lien vers une ancre, pour un style CSS de type ID, pour des manipulations en JavaScript, etc.
- `style` : cet attribut vous permet d'indiquer directement le code CSS à appliquer. Vous n'êtes donc pas obligés d'avoir une feuille de style à part, vous pouvez mettre directement les attributs CSS. Notez qu'il est préférable de ne pas utiliser cet attribut et de passer à la place par une feuille de style externe, car cela rend votre site plus facile à mettre à jour par la suite.

Ces trois attributs ne sont pas réservés aux balises génériques : vous pouvez aussi les utiliser sans aucun problème dans la plupart des autres balises.

Chapitre 22

Mémento des propriétés CSS

Difficulté :

Cette page est une liste *non exhaustive* des propriétés CSS qui existent en CSS3. Pour la plupart, ce sont des propriétés que nous avons vues dans le cours, mais vous trouverez aussi quelques nouvelles propriétés que nous n'avons pas abordées.

La liste est non exhaustive car mon but n'est pas de faire la liste de toutes les propriétés CSS qui peuvent exister : il y en a vraiment trop (plus de deux cents !) et certaines sont très rarement utilisées.



Propriétés de mise en forme du texte

Je résume ici la plupart des propriétés de **mise en forme du texte**.

Qu'est-ce que la mise en forme de texte ? C'est tout ce qui touche à la présentation du texte proprement dit : le gras, l'italique, le souligné, la police, l'alignement, etc.

Propriété	**Description**
`font-family`	Nom de police
`@font-face`	Police personnalisée
`font-size`	Taille du texte
`font-weight`	Gras
`font-style`	Italique
`text-decoration`	Soulignement, ligne au-dessus, barré ou clignotant
`font-variant`	Petites capitales
`text-transform`	Capitales
`font`	Super propriété de police. Combine : `font-weight`, `font-style`, `font-size`, `font-variant`, `font-family`.
`text-align`	Alignement horizontal
`vertical-align`	Alignement vertical (cellules de tableau ou éléments `inline-block` uniquement)
`line-height`	Hauteur de ligne
`text-indent`	Alinéa
`white-space`	Césure
`word-wrap`	Césure forcée
`text-shadow`	Ombre de texte

Propriétés de couleur et de fond

Propriété	**Description**
`color`	Couleur du texte
`background-color`	Couleur de fond
`background-image`	Image de fond
`background-attachment`	Fond fixe
`background-repeat`	Répétition du fond
`background-position`	Position du fond
`background`	Super propriété du fond. Combine : `background-image`, `background-repeat`, `background-attachment`, `background-position`
`opacity`	Transparence

Propriétés des boîtes

Propriété	Description
width	Largeur
height	Hauteur
min-width	Largeur minimale
max-width	Largeur maximale
min-height	Hauteur minimale
max-height	Hauteur maximale
margin-top	Marge en haut
margin-left	Marge à gauche
margin-right	Marge à droite
margin-bottom	Marge en bas
margin	Super-propriété de marge. Combine : margin-top, margin-right, margin-bottom, margin-left.
padding-top	Marge intérieure en haut
padding-left	Marge intérieure à gauche
padding-right	Marge intérieure à droite
padding-bottom	Marge intérieure en bas
padding	Super-propriété de marge intérieure. Combine : padding-top, padding-right, padding-bottom, padding-left.
border-width	Épaisseur de bordure
border-color	Couleur de bordure
border-style	Type de bordure
border	Super-propriété de bordure. Combine border-width, border-color, border-style. Existe aussi en version border-top, border-right, border-bottom, border-left.
border-radius	Bordure arrondie
box-shadow	Ombre de boîte

Propriétés de positionnement et d'affichage

Propriété	Description
display	Type d'élément (block, inline, inline-block, none...)
visibility	Visibilité
clip	Affichage d'une partie de l'élément
overflow	Comportement en cas de dépassement
float	Flottant
clear	Arrêt d'un flottant
position	Positionnement
top	Position par rapport au haut
bottom	Position par rapport au bas

left	Position par rapport à la gauche
right	Position par rapport à la droite
z-index	Ordre d'affichage en cas de superposition

Propriétés des listes

Propriété	Description
list-style-type	Type de liste
list-style-position	Position en retrait
list-style-image	Puce personnalisée
list-style	Super-propriété de liste. Combine list-style-type, list-style-position, list-style-image.

Propriétés des tableaux

Propriété	Description
border-collapse	Fusion des bordures
empty-cells	Affichage des cellules vides
caption-side	Position du titre du tableau

Autres propriétés

Propriété	Description
cursor	Curseur de souris

Chapitre 23

Gérer la compatibilité entre les navigateurs

Difficulté :

Voilà le scénario : vous travaillez d'arrache-pied sur le design de votre site web. Vous y passez du temps, vous le soignez, et au final vous êtes contents de vous. Vous faites tester le site par un ami et là il vous répond : « C'est un peu moche quand même ».

Soit vous et votre ami avez des goûts complètement différents, soit le site est joli *chez vous*, mais pas *chez lui*. En effet, les navigateurs n'ont pas exactement le même rendu, même s'ils s'efforcent de suivre les mêmes règles HTML et CSS. Plus le navigateur est ancien, plus il y a de risques qu'il ne comprenne pas les fonctionnalités que vous utilisez.

Plutôt que de céder à la panique, ce chapitre vous propose quelques méthodes pour apprendre à gérer ces différences entre navigateurs... et obtenir le meilleur résultat possible quel que soit le navigateur.



Les commentaires conditionnels

Souvent, ce sont les anciennes versions d'Internet Explorer qui vont vous poser problème. Comme certaines d'entre elles sont toujours utilisées, vous devez apprendre à composer avec.

La syntaxe des commentaires conditionnels

Heureusement, il existe une technique très pratique pour adapter votre code aux anciennes versions d'Internet Explorer. Grâce à des commentaires conditionnels, vous pouvez faire en sorte qu'une portion du code HTML soit lue uniquement par certaines versions d'IE. Ce sont des commentaires HTML qui ont une forme un peu spéciale et qui ne seront lus que par IE.

```
<!--[if IE]>
Code HTML réservé à IE
<![endif]-->
```

L'ensemble forme un commentaire HTML : il commence par `<!--` et se termine par `-->`. Les navigateurs ignoreront donc le code qui se trouve à l'intérieur... sauf IE, qui y voit une condition `[if IE]` signifiant « Si c'est Internet Explorer ».

L'intérêt est que cette technique permet de cibler différentes versions d'Internet Explorer. Par exemple :

```
<!--[if IE8]>
```

... ciblera uniquement Internet Explorer 8.

On peut utiliser les symboles suivants pour viser un ensemble de versions :

- `lt` : inférieur ;
- `lte` : inférieur ou égal ;
- `gt` : supérieur ;
- `gte` : supérieur ou égal.

Ainsi :

```
<!--[if lte IE8]>
```

... signifie « Toutes les versions d'Internet Explorer inférieures ou égales à IE8 » (c'est-à-dire IE8, IE7, IE6...). À l'inverse, on peut cibler les navigateurs autres qu'Internet Explorer avec la syntaxe suivante :

```
<!--[if !IE]><!-->
Code HTML pour les navigateurs différents d'IE
<!--<![endif]-->
```

La syntaxe a l'air un peu complexe, mais l'astuce est bien conçue. Si vous regardez bien, il y a deux balises de commentaires. Le code HTML se trouve en-dehors des balises de commentaires, il sera donc lu par tous les navigateurs... sauf IE, qui lit les commentaires lui demandant de ne pas lire le code HTML qui suit.

Utilité des commentaires conditionnels



À quoi servent les commentaires conditionnels ?

On peut utiliser les commentaires conditionnels pour afficher un message à destination des vieilles versions d'Internet Explorer : « Attention, vous naviguez avec un navigateur très ancien, mettez-vous à jour… ».

Cependant, on s'en sert surtout pour charger des feuilles de style spéciales pour IE :

```
<!--[if lte IE8]>
<link rel="stylesheet" href="style_ie.css" />
<![endif]-->
```

La feuille `style_ie.css` sera lue ici uniquement par les versions d'IE *inférieures ou égales* à IE8. À l'intérieur, on peut « adapter » le code CSS pour faire en sorte que le site s'affiche mieux sur ces navigateurs. À vous de modifier la dimension des éléments si nécessaire, d'utiliser d'autres méthodes de positionnement plus adaptées, etc.

Cependant, vu qu'il y a plusieurs versions d'Internet Explorer en circulation (IE6, IE7, IE8, IE9…), une feuille de style ne suffit pas. Parfois, on voudra effectuer une modification uniquement pour IE6 et IE7, parfois uniquement pour IE6, parfois pour IE8 et IE9… À moins de créer une feuille de style par version, on ne s'en sort pas. Mais cela fait alors beaucoup de fichiers `.css` à gérer.

La solution, régulièrement employée par les webmasters expérimentés aujourd'hui, consiste à donner un nom de classe à la balise `<body>` qui englobe le corps de la page. Ce nom de classe indiquera dans le fichier `.css` le nom de la version d'IE correspondant au navigateur. Dans ce cas, un seul fichier CSS suffit. Par exemple, si on écrit :

```
<body class="ie8">
```

… cela nous donnera dans le fichier CSS l'information que le navigateur est IE8. On peut ensuite, dans ce fichier CSS, écrire :

```
.ie8 nav
{
    /* Modification du style de la balise nav pour IE8 */
}
```

Avec les commentaires conditionnels, on peut écrire plusieurs versions de la balise `<body>` en fonction du navigateur :

```
<!--[if IE6]><body class="ie6 old_ie"><![endif]-->
<!--[if IE7]><body class="ie7 old_ie"><![endif]-->
<!--[if IE8]><body class="ie8"><![endif]-->
<!--[if IE9]><body class="ie9"><![endif]-->
<!--[if !IE]><!--><body><!--<![endif]-->
```



On peut donner plusieurs noms de classes à une balise, en les séparant par un espace comme on l'a fait ici. Ainsi, si le navigateur est IE6, la balise `<body>` aura deux classes : `ie6` et `old_ie`[1]. La classe `old_ie` permet donc de cibler simultanément IE6 et IE7.

Les hacks CSS

« *Hack* » signifie « bidouille » en anglais. Il s'agit d'astuces qui permettent de faire en sorte que les propriétés CSS soient lues par certaines versions d'Internet Explorer uniquement. Le principe est le même que les commentaires conditionnels, mais c'est moins « propre » : les hacks se basent sur des erreurs de ces navigateurs, qui lisent des propriétés CSS alors qu'ils ne devraient pas.



L'usage des hacks est globalement déconseillé car on se base sur des bugs des navigateurs. Rien ne garantit qu'ils fonctionneront à l'avenir.

Voici l'un des hacks les plus célèbres :

```
balise
{
    margin: 15px; /* Pour les autres navigateurs */
    ma\rgin: 20px; /* Pour IE6 et IE7 */
}
```

Un bug dans IE6 et IE7 leur fait lire la seconde propriété. Normalement, la présence de l'antislash (\) au milieu du nom de la propriété rend celle-ci invalide, mais ces navigateurs ignorent l'antislash et lisent la propriété comme si de rien n'était.

D'autres hacks existent, citons par exemple :

```
balise
{
    margin: 15px; /* Pour les autres navigateurs */
    *margin: 20px; /* Pour IE6 et IE7 */
    _margin: 20px; /* Pour IE6 */
}
```

La présence d'une étoile au début du nom de la propriété fait que tous les navigateurs, sauf IE6 et IE7, ignorent la ligne. De même, la présence d'un underscore (_) fait que la ligne est ignorée par tous les navigateurs, sauf IE6.

1. *old IE* signifie « vieil IE » en anglais.

Les outils de débogage

Parfois, votre page ne s'affiche pas comme vous le souhaitez, même sur un navigateur récent. Dans ce cas, il faut partir à la recherche de l'erreur que l'on a faite. Plusieurs techniques sont à votre disposition, je vais vous présenter celles que j'utilise le plus souvent.

La couleur de fond pour marquer les éléments

La technique la plus simple pour faire du débogage consiste tout simplement... à donner une couleur de fond à un élément qui ne se comporte pas comme vous le souhaitez (par exemple, un menu qui se positionne mal). C'est donc aussi simple que cela :

```
nav
{
    background-color: blue;
}
```

Cette couleur de fond, très voyante, est temporaire : vous vous en servez uniquement pour voir la « place » qu'occupe votre élément. Souvent, cela vous permet de comprendre ce qui se passe dans votre page.

Cependant, si cette technique ne suffit pas, il va falloir sortir l'artillerie lourde : un outil de débogage.

Les outils de débogage

Ce sont des outils très puissants (et assez impressionnants) qui permettent de manipuler la page web *directement* dans le navigateur. La plupart des navigateurs actuels sont équipés de ces outils, spécialement conçus pour les développeurs web que nous sommes :

- **Google Chrome** : allez dans le menu `Clé à molette` > `Outils` > `Outils de débogage` (figure 23.1). Vous pouvez aussi faire un clic droit sur un élément de votre page, puis cliquer sur `Inspecter l'élément`.
- **Mozilla Firefox** : installez l'extension *Firebug*. Un bouton apparaîtra en bas ou en haut à droite de la fenêtre (cela dépend de la version du navigateur) pour ouvrir le volet de débogage.
- **Internet Explorer** : allez dans le menu `Roue dentée > Outils de développement` (disponible à partir d'IE8).
- **Opera** : vous pouvez lancer son très puissant outil de débogage *Dragonfly* avec `Ctrl` + `Shift` + `I`. Vous pouvez aussi faire un clic droit sur un élément de votre page, puis cliquer sur `Inspecter l'élément`.

▷ Installer Firebug
Code web : 113479



Sur Google Chrome, Mozilla Firefox et Internet Explorer, vous pouvez ouvrir le volet de débogage en appuyant sur la touche `F12` de votre clavier. Sur Google Chrome et Firefox, vous pouvez également appuyer sur les touches `Ctrl` + `Shift` + `I` pour ouvrir le volet de débogage.



Figure 23.1 – L'outil de débogage de Google Chrome (en bas du navigateur)

Il y a plusieurs façons de se servir de ces outils. Vous pouvez étudier la hiérarchie de vos balises en affichant le code source (en bas à gauche sur la figure 23.1) et modifier en temps réel le code de votre page : faites un double-clic sur un texte ou une balise à modifier, et vous verrez le résultat changer dans la fenêtre du navigateur au fur et à mesure ! Attention toutefois : vos changements ne sont pas enregistrés, vous faites là un simple test pour comprendre le comportement du navigateur.

En bas à droite, vous avez la liste des propriétés CSS qui s'appliquent à l'élément actuellement sélectionné dans le code source. Vous pouvez lui ajouter des propriétés, en retirer, les modifier, etc. Tous les changements sont appliqués immédiatement. Notez que les propriétés CSS qui sont écrasées par d'autres propriétés sont barrées ou grisées : cela vous permet de comprendre comment votre feuille de style se comporte.

Enfin, pour cibler directement un élément de votre page, vous pouvez faire un clic droit sur l'élément à analyser, puis cliquer sur `Inspecter l'élément`. La barre de débogage propose aussi un bouton pour cibler un élément (une loupe sous Google Chrome, un curseur sous IE, etc.).



Ces outils proposent de nombreuses autres fonctionnalités plus avancées que je ne détaillerai pas ici mais que je vous laisse le loisir de découvrir : analyse du temps de chargement de la page, débogage JavaScript... Sachez cependant que, la plupart du temps, on s'en sert surtout pour faire du débogage du code HTML et CSS.

Index

Dépôt légal : décembre 2011
ISBN : 978-2-9535278-8-9
Code éditeur : 978-2-9535278
Imprimé en France

Achevé d'imprimer le 20 décembre 2011
sur les presses de Corlet Imprimeur (Condé-sur-Noireau)
Numéro imprimeur : 142357



Mentions légales :
Crédit photo Mathieu Nebra 4e de couverture : Xavier Granet - 2009
Conception couverture : Fan Jiyong
Illustrations chapitres : Fan Jiyong
Réalisation design du TP : Fan Jiyong

RÉALISEZ VOTRE SITE WEB AVEC HTML5 ET CSS3

Vous rêvez d'apprendre à créer des **sites web** mais vous avez peur que ce soit compliqué car vous débutez ?

Ce livre est fait pour vous ! **Conçu pour les débutants**, il vous permettra de découvrir **HTML5 et CSS3**, les dernières technologies en matière de **création de sites web**.



Plus de **20 chapitres** de difficulté progressive

Concevez le design d'un site de **A à Z**

Un cours pensé pour les débutants

- ► Aucun pré-requis, à part savoir allumer son ordinateur
- ► Une difficulté progressive pour ne perdre aucun lecteur en route
- ► Plébiscité par les professeurs et professionnels de l'informatique, mais aussi par leurs élèves !

Réalisez le site web dont vous avez toujours rêvé !

- ► Qu'est-ce qu'un site web et comment font les pros pour en créer ?
- ► Installez les logiciels dont vous avez besoin, que vous soyez sous Windows, Mac ou Linux
- ► Découvrez les balises HTML5 afin de structurer votre site
- ► Habillez vos pages web grâce à CSS3
- ► Réussissez la mise en page de votre site pas à pas à travers des travaux pratiques !
- ► Insérez facilement des images et vidéos sur votre site web
- ► Hébergez votre site sur le Web

À qui ce livre est-il destiné ?

- ► Aux passionnés d'informatique qui veulent aller plus loin avec leur ordinateur
- ► Aux étudiants dans le domaine des nouvelles technologies qui recherchent un support de cours
- ► À toutes les personnes qui désirent créer des sites web

À propos de l'auteur



Mathieu Nebra alias M@teo21
Fondateur du Site du Zéro

Jeune passionné de nouvelles technologies, il cherche en vain en librairie des cours accessibles aux débutants pour se former.

Afin de prouver que l'on peut faire « plus clair et plus simple » il crée le Site du Zéro, aujourd'hui devenu la référence des cours pour débutants en ligne avec plusieurs millions de visites par mois.

Ses précédents ouvrages sur la programmation sont aujourd'hui des best-sellers et ont permis à de nombreux débutants de se former sur le C, PHP, Linux, C++…

Ce livre est issu du Site du Zéro

Retrouvez dans ce livre les cours du Site du Zéro dans une édition revue et corrigée avec de nouveaux chapitres inédits du même auteur !

Téléchargez les codes sources ou accédez aux exemples en ligne grâce aux « codes web » inclus dans ce livre.

ISBN : 978-2-9535278-8-9



Prix public : 25 € TTC



www.siteduzero.com

